made with doc-o-matic

RAD Studio VCL Reference

Copyright(C) 2009 Embarcadero Technologies, Inc. All Rights Reserved.

Table of Contents

IB Namespace 1

Classes **3**

EIBClientError Class 3

EIBError Class 3

EIBError Members 4

Create Constructor 4

EIBError Properties 5

EIBInterBaseError Class 5

EIBInterBaseRoleError Class 6

Functions **7**

IB.CheckStatusVector Function 7

IB.GetIBDataBaseErrorMessages Function 7

IB.IBAlloc Function 8

IB.IBDataBaseError Function 8

IB.IBError Function 8

IB.SetIBDataBaseErrorMessages Function 9

IB.StatusVector Function 9

IB.StatusVectorArray Function 9

IB.StatusVectorAsText Function 10

Structs, Records, Enums **11**

IB.TIBClientError Enumeration 11

IB.TIBDataBaseErrorMessage Enumeration 15

IB.TTraceFlag Enumeration 15

Types **17**

IB.PStatusVector Type 17

IB.TIBDataBaseErrorMessages Type 17

IB.TStatusVector Type 18

IB.TTraceFlags Type 18

Variables **20**

IB.IBCS Variable 20

Constants **21**

IB.IBBigLocalBufferLength Constant 21

IB.IBErrorMessages Constant 21

IB.IBHugeLocalBufferLength Constant 22
IB.IBLocalBufferLength Constant 23
IB.IBPalette1 Constant 23
IB.IBPalette2 Constant 23

IBBatchMove Namespace 25

Classes **26**
TIBBatchMove Class 26
TIBBatchMove Members 26
TIBBatchMove.Create Constructor 27
TIBBatchMove.Destroy Destructor 28
TIBBatchMove Methods 28
TIBBatchMove Properties 29

Structs, Records, Enums **38**
IBBatchMove.TBatchMode Enumeration 38

IBBlob Namespace 39

Classes **40**
TIBBlobStream Class 40
TIBBlobStream Members 40
TIBBlobStream.Create Constructor 41
TIBBlobStream.Destroy Destructor 42
TIBBlobStream Methods 42
TIBBlobStream Properties 50

Functions **54**
IBBlob.GetBlobInfo Function 54
IBBlob.ReadBlob Function 54
IBBlob.WriteBlob Function 55

Constants **56**
IBBlob.DefaultBlobSegmentSize Constant 56

IBCustomDataSet Namespace 57

Classes **59**
TIBBCDField Class 59
TIBBCDField Members 59
TIBBCDField.Create Constructor 60

TIBBCDField Methods 60
TIBBCDField Properties 61
TIBCustomDataSet Class 62
TIBCustomDataSet Members 62
TIBCustomDataSet.Create Constructor 66
TIBCustomDataSet.Destroy Destructor 66
TIBCustomDataSet Methods 67
TIBCustomDataSet Properties 92
TIBCustomDataSet Events 108
TIBDataLink Class 111
TIBDataLink Members 112
TIBDataLink.Create Constructor 112
TIBDataLink.Destroy Destructor 112
TIBDataLink Methods 113
TIBDataSet Class 114
TIBDataSet Members 114
TIBDataSet Methods 118
TIBDataSet Properties 121
TIBDataSet Events 133
TIBDataSetUpdateObject Class 134
TIBDataSetUpdateObject Members 135
TIBDataSetUpdateObject.Create Constructor 135
TIBDataSetUpdateObject.Destroy Destructor 136
TIBDataSetUpdateObject Methods 136
TIBDataSetUpdateObject Properties 137
TIBDSBlobStream Class 138
TIBDSBlobStream Members 138
TIBDSBlobStream.Create Constructor 139
TIBDSBlobStream.Destroy Destructor 139
TIBDSBlobStream Fields 139
TIBDSBlobStream Methods 140
TIBGeneratorField Class 142
TIBGeneratorField Members 142
TIBGeneratorField.Create Constructor 143
TIBGeneratorField Methods 143
TIBGeneratorField Properties 144
TIBStringField Class 146
TIBStringField Members 146
TIBStringField.Create Constructor 147

TIBStringField Methods 147

Structs, Records, Enums **149**

IBCustomDataSet.TCachedUpdateStatus Enumeration 149

IBCustomDataSet.TFieldData Record 150

IBCustomDataSet.TIBDBKey Record 150

IBCustomDataSet.TIBGeneratorApplyEvent Enumeration 151

IBCustomDataSet.TIBUpdateAction Enumeration 151

IBCustomDataSet.TLiveMode Enumeration 152

IBCustomDataSet.TRecordData Record 153

Types **154**

IBCustomDataSet.PBlobDataArray Type 154

IBCustomDataSet.PDateTime Type 154

IBCustomDataSet.PFieldData Type 155

IBCustomDataSet.PIBDBKey Type 155

IBCustomDataSet.PRecordData Type 155

IBCustomDataSet.TBlobDataArray Type 156

IBCustomDataSet.TIBUpdateErrorEvent Type 156

IBCustomDataSet.TIBUpdateRecordEvent Type 157

IBCustomDataSet.TIBUpdateRecordTypes Type 157

IBCustomDataSet.TLiveModes Type 157

Variables **159**

IBCustomDataSet.CreateProviderProc Variable 159

Constants **160**

IBCustomDataSet.BufferCacheSize Constant 160

IBCustomDataSet.DefaultFieldClasses Constant 160

IBCustomDataSet.UniCache Constant 161

IBDatabase Namespace 163

Classes **165**

TIBBase Class 165

TIBBase Members 165

TIBBase.Create Constructor 166

TIBBase.Destroy Destructor 167

TIBBase Fields 167

TIBBase Methods 170

TIBBase Properties 173

TIBBase Events 174

TIBDataBase Class 176
TIBDataBase Members 177
TIBDataBase.Create Constructor 178
TIBDataBase.Destroy Destructor 178
TIBDataBase Methods 179
TIBDataBase Properties 188
TIBDataBase Events 198
TIBSchema Class 201
TIBSchema Members 201
TIBSchema Methods 202
TIBTimer Class 202
TIBTimer Members 203
TIBTimer.Create Constructor 203
TIBTimer.Destroy Destructor 204
TIBTimer Methods 205
TIBTimer Properties 205
TIBTimer Events 206
TIBTransaction Class 206
TIBTransaction Members 207
TIBTransaction.Create Constructor 208
TIBTransaction.Destroy Destructor 208
TIBTransaction Methods 208
TIBTransaction Properties 215
TIBTransaction Events 219

Interfaces **221**
IIBEventNotifier Interface 221
IIBEventNotifier Members 221
IIBEventNotifier Methods 222

Functions **224**
IBDatabase.GenerateDPB Function 224
IBDatabase.GenerateTPB Function 224

Structs, Records, Enums **225**
IBDatabase.TAutoStopAction Enumeration 225
IBDatabase.TTransactionAction Enumeration 226

Types **227**
IBDatabase.TIBDatabaseLoginEvent Type 227
IBDatabase.TIBFileName Type 227

Constants **229**
IBDatabase.DPBConstantNames Constant 229
IBDatabase.DPBPrefix Constant 230
IBDatabase.TPBConstantNames Constant 230
IBDatabase.TPBPrefix Constant 230

IBDatabaseInfo Namespace 231

Classes **232**
TIBDatabaseInfo Class 232
TIBDatabaseInfo Members 232
TIBDatabaseInfo.Create Constructor 234
TIBDatabaseInfo.Destroy Destructor 234
TIBDatabaseInfo Fields 235
TIBDatabaseInfo Methods 237
TIBDatabaseInfo Properties 246

IBEvents Namespace 257

Classes **258**
TIBEvents Class 258
TIBEvents Members 258
TIBEvents.Create Constructor 259
TIBEvents.Destroy Destructor 260
TIBEvents Methods 260
TIBEvents Properties 263
TIBEvents Events 265

Types **266**
IBEvents.TErrorEvent Type 266
IBEvents.TEventAlert Type 266

IBExtract Namespace 269

Classes **271**
TIBExtract Class 271
TIBExtract Members 271
TIBExtract.Create Constructor 272
TIBExtract.Destroy Destructor 273
TIBExtract Methods 273

TIBExtract Properties 281

Structs, Records, Enums **283**

IBExtract.TExtractObjectTypes Enumeration 283

IBExtract.TExtractType Enumeration 284

IBExtract.TPrivTypes Record 285

IBExtract.TSQLType Record 286

Types **287**

IBExtract.TExtractTypes Type 287

IBExtract.TSQLTypes Type 288

Constants **289**

IBExtract.ColumnTypes Constant 290

IBExtract.FILE_cache Constant 290

IBExtract.FILE_conditional Constant 290

IBExtract.FILE_inactive Constant 291

IBExtract.FILE_manual Constant 291

IBExtract.FILE_shadow Constant 291

IBExtract.IntegralSubtypes Constant 292

IBExtract.LOG_default Constant 292

IBExtract.LOG_overflow Constant 292

IBExtract.LOG_raw Constant 293

IBExtract.LOG_serial Constant 293

IBExtract.MAX_INTSUBTYPES Constant 293

IBExtract.MAXSUBTYPES Constant 294

IBExtract.obj_computed Constant 294

IBExtract.obj_count Constant 294

IBExtract.obj_exception Constant 295

IBExtract.obj_expression_index Constant 295

IBExtract.obj_field Constant 295

IBExtract.obj_index Constant 296

IBExtract.obj_procedure Constant 296

IBExtract.obj_relation Constant 296

IBExtract.obj_sql_role Constant 297

IBExtract.obj_trigger Constant 297

IBExtract.obj_user Constant 297

IBExtract.obj_user_group Constant 298

IBExtract.obj_validation Constant 298

IBExtract.obj_view Constant 298

IBExtract.ODS_VERSION10 Constant 299

IBExtract.ODS_VERSION6 Constant 299
IBExtract.ODS_VERSION7 Constant 299
IBExtract.ODS_VERSION8 Constant 300
IBExtract.ODS_VERSION9 Constant 300
IBExtract.priv_DELETE Constant 300
IBExtract.priv_EXECUTE Constant 301
IBExtract.priv_INSERT Constant 301
IBExtract.priv_REFERENCES Constant 301
IBExtract.priv_SELECT Constant 302
IBExtract.priv_UNKNOWN Constant 302
IBExtract.priv_UPDATE Constant 302
IBExtract.PrivTypes Constant 303
IBExtract.SubTypes Constant 303
IBExtract.TriggerTypes Constant 303

IBInstall Namespace 305

Classes 307
EIBInstall Class 307
EIBInstall Members 307
Create Constructor 308
EIBInstall Properties 308
EIBInstallerError Class 309
EIBInstallError Class 310
TIBInstall Class 310
TIBInstall Members 310
TIBInstall.Create Constructor 311
TIBInstall.Destroy Destructor 312
TIBInstall Methods 312
TIBInstall Properties 315
TIBSetup Class 317
TIBSetup Members 317
TIBSetup.Create Constructor 318
TIBSetup Methods 318
TIBSetup Properties 319
TIBSetup Events 321
TIBUnInstall Class 322
TIBUnInstall Members 323
TIBUnInstall Methods 323

TIBUnInstall Properties 324
TInstallOptions Class 325
TInstallOptions Members 325
TInstallOptions Properties 325

Structs, Records, Enums 328
IBInstall.TCmdOption Enumeration 328
IBInstall.TConnectivityOption Enumeration 329
IBInstall.TErrorResult Enumeration 329
IBInstall.TExamplesOption Enumeration 330
IBInstall.TIBInstallerError Enumeration 330
IBInstall.TMainOption Enumeration 331
IBInstall.TStatusResult Enumeration 332
IBInstall.TWarningResult Enumeration 332

Types 333
IBInstall.TCmdOptions Type 333
IBInstall.TConnectivityOptions Type 333
IBInstall.TExamplesOptions Type 334
IBInstall.TIBSetupOnError Type 334
IBInstall.TIBSetupOnStatus Type 334
IBInstall.TIBSetupOnWarning Type 335
IBInstall.TIscError Type 335
IBInstall.TMainOptions Type 336

IBIntf Namespace 337

Functions 340
IBIntf.CheckIBInstallLoaded Function 340
IBIntf.CheckIBLoaded Function 341
IBIntf.FreeIBInstallLibrary Function 341
IBIntf.FreeIBLibrary Function 341
IBIntf.GetIBClientVersion Function 342
IBIntf.isc_decode_sql_date_stub Function 342
IBIntf.isc_decode_sql_time_stub Function 342
IBIntf.isc_decode_timestamp_stub Function 343
IBIntf.isc_encode_sql_date_stub Function 343
IBIntf.isc_encode_sql_time_stub Function 343
IBIntf.isc_encode_timestamp_stub Function 344
IBIntf.isc_install_clear_options_stub Function 344
IBIntf.isc_install_execute_stub Function 344

IBIntf.isc_install_get_info_stub Function 345
IBIntf.isc_install_get_message_stub Function 345
IBIntf.isc_install_load_external_text_stub Function 345
IBIntf.isc_install_precheck_stub Function 346
IBIntf.isc_install_set_option_stub Function 346
IBIntf.isc_install_unset_option_stub Function 346
IBIntf.isc_rollback_retaining_stub Function 347
IBIntf.isc_service_attach_stub Function 347
IBIntf.isc_service_detach_stub Function 347
IBIntf.isc_service_query_stub Function 348
IBIntf.isc_service_start_stub Function 348
IBIntf.isc_uninstall_execute_stub Function 348
IBIntf.isc_uninstall_precheck_stub Function 349
IBIntf.LoadIBInstallLibrary Function 349
IBIntf.LoadIBLibrary Function 349
IBIntf.TryIBLoad Function 350

Variables **351**

IBIntf.BLOB_get Variable 352
IBIntf.BLOB_put Variable 352
IBIntf.isc_add_user Variable 353
IBIntf.isc_attach_database Variable 353
IBIntf.isc_blob_info Variable 353
IBIntf.isc_cancel_events Variable 354
IBIntf.isc_close_blob Variable 354
IBIntf.isc_commit_retaining Variable 354
IBIntf.isc_commit_transaction Variable 355
IBIntf.isc_create_blob2 Variable 355
IBIntf.isc_database_info Variable 355
IBIntf.isc_decode_date Variable 356
IBIntf.isc_decode_sql_date Variable 356
IBIntf.isc_decode_sql_time Variable 356
IBIntf.isc_decode_timestamp Variable 357
IBIntf.isc_delete_user Variable 357
IBIntf.isc_detach_database Variable 357
IBIntf.isc_drop_database Variable 358
IBIntf.isc_dsql_alloc_statement2 Variable 358
IBIntf.isc_dsql_describe Variable 358
IBIntf.isc_dsql_describe_bind Variable 359
IBIntf.isc_dsql_execute Variable 359

IBIntf.isc_dsql_execute_immediate Variable 359
IBIntf.isc_dsql_execute2 Variable 360
IBIntf.isc_dsql_fetch Variable 360
IBIntf.isc_dsql_free_statement Variable 360
IBIntf.isc_dsql_prepare Variable 361
IBIntf.isc_dsql_set_cursor_name Variable 361
IBIntf.isc_dsql_sql_info Variable 361
IBIntf.isc_encode_date Variable 362
IBIntf.isc_encode_sql_date Variable 362
IBIntf.isc_encode_sql_time Variable 362
IBIntf.isc_encode_timestamp Variable 363
IBIntf.isc_event_block Variable 363
IBIntf.isc_event_counts Variable 363
IBIntf.isc_free Variable 364
IBIntf.isc_get_segment Variable 364
IBIntf.isc_install_clear_options Variable 364
IBIntf.isc_install_execute Variable 365
IBIntf.isc_install_get_info Variable 365
IBIntf.isc_install_get_message Variable 365
IBIntf.isc_install_load_external_text Variable 366
IBIntf.isc_install_precheck Variable 366
IBIntf.isc_install_set_option Variable 366
IBIntf.isc_install_unset_option Variable 367
IBIntf.isc_interprete Variable 367
IBIntf.isc_modify_user Variable 367
IBIntf.isc_open_blob2 Variable 368
IBIntf.isc_prepare_transaction Variable 368
IBIntf.isc_prepare_transaction2 Variable 368
IBIntf.isc_put_segment Variable 369
IBIntf.isc_que_events Variable 369
IBIntf.isc_rollback_retaining Variable 369
IBIntf.isc_rollback_transaction Variable 370
IBIntf.isc_service_attach Variable 370
IBIntf.isc_service_detach Variable 370
IBIntf.isc_service_query Variable 371
IBIntf.isc_service_start Variable 371
IBIntf.isc_sql_interprete Variable 371
IBIntf.isc_sqlcode Variable 372
IBIntf.isc_start_multiple Variable 372

IBIntf.isc_uninstall_execute Variable 372
IBIntf.isc_uninstall_precheck Variable 373
IBIntf.isc_vax_integer Variable 373

IBQuery Namespace 375

Classes 376
TIBQuery Class 376
TIBQuery Members 376
TIBQuery.Create Constructor 381
TIBQuery.Destroy Destructor 381
TIBQuery Methods 381
TIBQuery Properties 387
TIBQuery Events 394

IBServices Namespace 397

Classes 399
TConfigFileData Class 399
TConfigFileData Members 400
TConfigFileData.Create Constructor 400
TConfigFileData.Destroy Destructor 400
TConfigFileData Fields 400
TConfigParams Class 401
TConfigParams Members 401
TConfigParams.Create Constructor 402
TConfigParams.Destroy Destructor 402
TConfigParams Fields 402
TDatabaseInfo Class 404
TDatabaseInfo Members 404
TDatabaseInfo.Create Constructor 405
TDatabaseInfo.Destroy Destructor 405
TDatabaseInfo Fields 405
TIBBackupRestoreService Class 406
TIBBackupRestoreService Members 406
TIBBackupRestoreService Properties 407
TIBBackupService Class 408
TIBBackupService Members 408
TIBBackupService.Create Constructor 410
TIBBackupService.Destroy Destructor 410

TIBBackupService Methods 410
TIBBackupService Properties 410
TIBConfigService Class 412
TIBConfigService Members 412
TIBConfigService Methods 413
TIBConfigService Properties 417
TIBControlAndQueryService Class 417
TIBControlAndQueryService Members 418
TIBControlAndQueryService.Create Constructor 419
TIBControlAndQueryService Methods 419
TIBControlAndQueryService Properties 420
TIBControlService Class 421
TIBControlService Members 421
TIBControlService.Create Constructor 422
TIBControlService Methods 423
TIBControlService Properties 424
TIBCustomService Class 425
TIBCustomService Members 425
TIBCustomService.Create Constructor 426
TIBCustomService.Destroy Destructor 426
TIBCustomService Methods 427
TIBCustomService Properties 429
TIBCustomService Events 432
TIBLicensingService Class 433
TIBLicensingService Members 434
TIBLicensingService Methods 435
TIBLicensingService Properties 436
TIBLogService Class 437
TIBLogService Members 437
TIBLogService Methods 438
TIBRestoreService Class 439
TIBRestoreService Members 439
TIBRestoreService.Create Constructor 440
TIBRestoreService.Destroy Destructor 441
TIBRestoreService Methods 441
TIBRestoreService Properties 441
TIBSecurityService Class 443
TIBSecurityService Members 444
TIBSecurityService.Create Constructor 445

TIBSecurityService.Destroy Destructor 445
TIBSecurityService Methods 446
TIBSecurityService Properties 448
TIBServerProperties Class 452
TIBServerProperties Members 452
TIBServerProperties.Create Constructor 453
TIBServerProperties.Destroy Destructor 453
TIBServerProperties Methods 454
TIBServerProperties Properties 456
TIBStatisticalService Class 458
TIBStatisticalService Members 458
TIBStatisticalService Methods 459
TIBStatisticalService Properties 460
TIBValidationService Class 461
TIBValidationService Members 461
TIBValidationService.Create Constructor 462
TIBValidationService.Destroy Destructor 463
TIBValidationService Methods 463
TIBValidationService Properties 464
TLicenseInfo Class 466
TLicenseInfo Members 466
TLicenseInfo.Create Constructor 467
TLicenseInfo.Destroy Destructor 467
TLicenseInfo Fields 467
TLicenseMaskInfo Class 468
TLicenseMaskInfo Members 469
TLicenseMaskInfo Fields 469
TLimboTransactionInfo Class 469
TLimboTransactionInfo Members 470
TLimboTransactionInfo Fields 470
TUserInfo Class 472
TUserInfo Members 473
TUserInfo Fields 473
TVersionInfo Class 475
TVersionInfo Members 475
TVersionInfo Fields 476

Structs, Records, Enums 477

IBServices.TBackupOption Enumeration 477
IBServices.TLicensingAction Enumeration 478

IBServices.TOutputBufferOption Enumeration 478
IBServices.TPropertyOption Enumeration 479
IBServices.TProtocol Enumeration 479
IBServices.TRestoreOption Enumeration 480
IBServices.TSecurityAction Enumeration 480
IBServices.TSecurityModifyParam Enumeration 481
IBServices.TShutdownMode Enumeration 481
IBServices.TStatOption Enumeration 482
IBServices.TTransactionAction Enumeration 482
IBServices.TTransactionAdvise Enumeration 483
IBServices.TTransactionGlobalAction Enumeration 483
IBServices.TTransactionState Enumeration 484
IBServices.TValidateOption Enumeration 485

Types **486**
IBServices.TBackupOptions Type 486
IBServices.TLoginEvent Type 487
IBServices.TPropertyOptions Type 487
IBServices.TRestoreOptions Type 488
IBServices.TSecurityModifyParams Type 489
IBServices.TStatOptions Type 489
IBServices.TValidateOptions Type 490

Constants **491**
IBServices.DefaultBufferSize Constant 491
IBServices.SPBConstantNames Constant 491
IBServices.SPBConstantValues Constant 492
IBServices.SPBPrefix Constant 492

IBSQL Namespace 493

Classes **494**
TIBBatch Class 494
TIBBatch Members 494
TIBBatch Fields 495
TIBBatch Methods 496
TIBBatch Properties 496
TIBBatchInput Class 497
TIBBatchInput Members 498
TIBBatchInput Methods 498
TIBBatchOutput Class 498

TIBBatchOutput Members 499
TIBBatchOutput Methods 499
TIBInputDelimitedFile Class 500
TIBInputDelimitedFile Members 500
TIBInputDelimitedFile.Destroy Destructor 501
TIBInputDelimitedFile Fields 501
TIBInputDelimitedFile Methods 503
TIBInputDelimitedFile Properties 504
TIBInputRawFile Class 505
TIBInputRawFile Members 505
TIBInputRawFile.Destroy Destructor 506
TIBInputRawFile Fields 506
TIBInputRawFile Methods 506
TIBOutputDelimitedFile Class 507
TIBOutputDelimitedFile Members 508
TIBOutputDelimitedFile.Destroy Destructor 508
TIBOutputDelimitedFile Fields 508
TIBOutputDelimitedFile Methods 509
TIBOutputDelimitedFile Properties 510
TIBOutputRawFile Class 511
TIBOutputRawFile Members 512
TIBOutputRawFile.Destroy Destructor 512
TIBOutputRawFile Fields 512
TIBOutputRawFile Methods 513
TIBSQL Class 513
TIBSQL Members 514
TIBSQL.Create Constructor 516
TIBSQL.Destroy Destructor 516
TIBSQL Fields 516
TIBSQL Methods 520
TIBSQL Properties 529
TIBSQL Events 535
TIBXSQLDA Class 536
TIBXSQLDA Members 536
TIBXSQLDA.Create Constructor 537
TIBXSQLDA.Destroy Destructor 537
TIBXSQLDA Fields 538
TIBXSQLDA Methods 539
TIBXSQLDA Properties 542

TIBXSQLVAR Class 544
TIBXSQLVAR Members 545
TIBXSQLVAR.Create Constructor 545
TIBXSQLVAR Methods 546
TIBXSQLVAR Properties 548

Structs, Records, Enums **556**
IBSQL.TIBSQLTypes Enumeration 556

Types **558**
IBSQL.TIBXSQLVARArray Type 558

IBSQLMonitor Namespace 559

Classes **561**
TIBCustomSQLMonitor Class 561
TIBCustomSQLMonitor Members 561
TIBCustomSQLMonitor.Create Constructor 562
TIBCustomSQLMonitor.Destroy Destructor 562
TIBCustomSQLMonitor Methods 562
TIBCustomSQLMonitor Properties 563
TIBCustomSQLMonitor Events 564
TIBSQLMonitor Class 564
TIBSQLMonitor Members 565
TIBSQLMonitor Properties 565
TIBSQLMonitor Events 566

Interfaces **567**
IIBSQLMonitorHook Interface 567
IIBSQLMonitorHook Members 567
IIBSQLMonitorHook Methods 568
IIBSQLMonitorHook Properties 573

Functions **575**
IBSQLMonitor.DisableMonitoring Function 575
IBSQLMonitor.EnableMonitoring Function 575
IBSQLMonitor.MonitorHook Function 576
IBSQLMonitor.MonitoringEnabled Function 576

Types **577**
IBSQLMonitor.TSQLEvent Type 577

Constants **578**

IBSQLMonitor.WM_IBSQL_SQL_EVENT Constant 578
IBSQLMonitor.WM_MAX_IBSQL_MONITOR Constant 578
IBSQLMonitor.WM_MIN_IBSQL_MONITOR Constant 578

IBStoredProc Namespace 581

Classes 582
TIBStoredProc Class 582
TIBStoredProc Members 582
TIBStoredProc.Create Constructor 586
TIBStoredProc.Destroy Destructor 587
TIBStoredProc Methods 587
TIBStoredProc Properties 592
TIBStoredProc Events 595

IBTable Namespace 599

Classes 600
TIBTable Class 600
TIBTable Members 600
TIBTable.Create Constructor 605
TIBTable.Destroy Destructor 605
TIBTable Methods 605
TIBTable Properties 614
TIBTable Events 623

Structs, Records, Enums 626
IBTable.TIBTableType Enumeration 626

Types 627
IBTable.TIBTableTypes Type 627
IBTable.TIndexName Type 627

IBUpdateSQL Namespace 629

Classes 630
TIBUpdateSQL Class 630
TIBUpdateSQL Members 630
TIBUpdateSQL.Create Constructor 631
TIBUpdateSQL.Destroy Destructor 631
TIBUpdateSQL Methods 632

TIBUpdateSQL Properties 634

IBUtils Namespace 639

Functions 640

IBUtils.AddIBParamSQLForDetail Function 640
IBUtils.ExtractIdentifier Function 640
IBUtils.FormatIdentifier Function 641
IBUtils.FormatIdentifierValue Function 641
IBUtils.Max Function 641
IBUtils.Min Function 642
IBUtils.QuoteIdentifier Function 642
IBUtils.RandomInteger Function 642
IBUtils.RandomString Function 643
IBUtils.StripString Function 643

Constants 644

IBUtils.CR Constant 644
IBUtils.CRLF Constant 644
IBUtils.LF Constant 645
IBUtils.NULL_TERMINATOR Constant 645
IBUtils.TAB Constant 645

ImgList Namespace 647

Classes 648

TChangeLink Class 648
TChangeLink Members 648
TChangeLink.Destroy Destructor 649
TChangeLink Methods 649
TChangeLink Properties 650
TChangeLink Events 650
TCustomImageList Class 651
TCustomImageList Members 651
TCustomImageList.Create Constructor 653
TCustomImageList.CreateSize Constructor 653
TCustomImageList.Destroy Destructor 654
TCustomImageList Methods 654
TCustomImageList Properties 675
TCustomImageList Events 684

Structs, Records, Enums **685**
ImgList.TColorDepth Enumeration 685
ImgList.TDrawingStyle Enumeration 686
ImgList.TImageType Enumeration 687
ImgList.TLoadResource Enumeration 687
ImgList.TResType Enumeration 688

Types **689**
ImgList.TImageIndex Type 689
ImgList.TImageListHandle Type 689
ImgList.TLoadResources Type 689
ImgList.TOverlay Type 690

IMouse Namespace 691

Classes **693**
TPanningWindow Class 693
TPanningWindow Members 693
TPanningWindow.Create Constructor 694
TPanningWindow.Destroy Destructor 694
TPanningWindow Methods 695
TPanningWindow Properties 703
TPanningWindow Events 704

Functions **706**
IMouse.StartPan Function 706

Structs, Records, Enums **707**
IMouse.TPanDirection Enumeration 707
IMouse.TPanOption Enumeration 707

Types **709**
IMouse.TPanDirections Type 709
IMouse.TPanOptions Type 709
IMouse.TUpdateEvent Type 709

Constants **711**
IMouse.crPanAll Constant 711
IMouse.crPanDown Constant 711
IMouse.crPanDownLeft Constant 712
IMouse.crPanDownRight Constant 712
IMouse.crPanLeft Constant 712

IMouse.crPanLeftRight Constant 713
IMouse.crPanRight Constant 713
IMouse.crPanUp Constant 713
IMouse.crPanUpDown Constant 714
IMouse.crPanUpLeft Constant 714
IMouse.crPanUpRight Constant 714

IniFiles Namespace 715

Classes 716
EIniFileException Class 716
TCustomIniFile Class 716
TCustomIniFile Members 720
TCustomIniFile.Create Constructor 720
TCustomIniFile Methods 721
TCustomIniFile Properties 749
THashedStringList Class 749
THashedStringList Members 750
THashedStringList.Destroy Destructor 750
THashedStringList Methods 751
TIniFile Class 752
TIniFile Members 755
TIniFile.Destroy Destructor 756
TIniFile Methods 757
TMemIniFile Class 765
TMemIniFile Members 769
Create Constructor 770
TMemIniFile.Destroy Destructor 770
TMemIniFile Methods 771
TMemIniFile Properties 782
TStringHash Class 783
TStringHash Members 783
TStringHash.Create Constructor 783
TStringHash.Destroy Destructor 784
TStringHash Methods 784

Structs, Records, Enums 787
IniFiles.THashItem Record 787

Types 788
IniFiles.PHashItem Type 788

IniFiles.PPHashItem Type 788

IntfInfo Namespace 789

Classes **791**

EInterfaceRTTIException Class 791

Functions **792**

IntfInfo.GetClsMemberTypeInfo Function 792
IntfInfo.GetDynArrayElTypeInfo Function 792
IntfInfo.GetDynArrayNextInfo Function 793
IntfInfo.GetDynArrayNextInfo2 Function 793
IntfInfo.GetIntfMetaData Function 793
IntfInfo.GetIntfMetaData Function 794
IntfInfo.GetMethNum Function 794
IntfInfo.GetPropListFlat Function 794
IntfInfo.IsStoredPropConst Function 795
IntfInfo.OtherTypeName Function 795
IntfInfo.SameTypeInfo Function 795
IntfInfo.TypeNamesMatch Function 796

Structs, Records, Enums **797**

IntfInfo.TCallConv Enumeration 797
IntfInfo.TFillMethodArrayOpt Enumeration 797
IntfInfo.TIntfMetaData Record 798
IntfInfo.TIntfMethEntry Record 799
IntfInfo.TIntfParamEntry Record 800

Types **801**

IntfInfo.PIntfMetaData Type 801
IntfInfo.PIntfMethEntry Type 801
IntfInfo.PIntfParamEntry Type 802
IntfInfo.TIntfMethEntryArray Type 802
IntfInfo.TIntfParamEntryArray Type 802
IntfInfo.TPIntfMethEntryArray Type 803

Constants **804**

IntfInfo.CallingConventionName Constant 804
IntfInfo.TypeInfoNames Constant 804

InvokeRegistry Namespace 807

Classes **810**
ERemotableException Class 810
ERemotableException Members 811
Create Constructor 811
ERemotableException Properties 812
ETypeRegistryException Class 813
TDataContext Class 814
TDataContext Members 815
TDataContext.Create Constructor 815
TDataContext.Destroy Destructor 815
TDataContext Fields 816
TDataContext Methods 818
THeaderList Class 823
THeaderList Members 823
THeaderList.Create Constructor 824
THeaderList.Destroy Destructor 824
THeaderList Methods 825
THeaderList Properties 827
TInvContext Class 829
TInvContext Members 830
TInvContext Methods 831
TInvokableClass Class 833
TInvokableClass Members 834
TInvokableClass.Create Constructor 835
TInvokableClass Fields 835
TInvokableClass Methods 836
TInvokableClass Properties 839
TInvokableClassRegistry Class 840
TInvokableClassRegistry Members 840
TInvokableClassRegistry.Create Constructor 842
TInvokableClassRegistry.Destroy Destructor 842
TInvokableClassRegistry Methods 842
TRemotable Class 868
TRemotable Members 869
TRemotable.Create Constructor 870
TRemotable.Destroy Destructor 870
TRemotable Fields 871
TRemotable Methods 871
TRemotable Properties 873

TRemotableTypeRegistry Class 874
TRemotableTypeRegistry Members 875
TRemotableTypeRegistry.Create Constructor 876
TRemotableTypeRegistry.Destroy Destructor 876
TRemotableTypeRegistry Methods 876
TRemotableTypeRegistry Properties 891
TRemotableXS Class 892
TRemotableXS Members 893
TRemotableXS Methods 893
TSOAPAttachment Class 895
TSOAPAttachment Members 897
TSOAPAttachment.Create Constructor 897
TSOAPAttachment.Destroy Destructor 898
TSOAPAttachment Methods 898
TSOAPAttachment Properties 904
TSOAPHeader Class 908
TSOAPHeader Members 909
TSOAPHeader Methods 910
TSOAPHeader Properties 911
TSOAPHeaders Class 912
TSOAPHeaders Members 913
TSOAPHeaders Methods 913
TSOAPHeadersBase Class 916
TSOAPHeadersBase Members 917
TSOAPHeadersBase Fields 917
TSOAPHeadersBase Methods 918

Interfaces 919

IHeadersSetter Interface 919
IHeadersSetter Members 919
IHeadersSetter Methods 919
IObjConverter Interface 920
IObjConverter Members 921
IObjConverter Methods 921
ISOAPHeaders Interface 923
ISOAPHeaders Members 924
ISOAPHeaders Methods 924
ISOAPHeaders Properties 927

Functions 929

InvokeRegistry.GetRemotableDataContext Function 929
InvokeRegistry.InvRegistry Function 930
InvokeRegistry.RemClassRegistry Function 930
InvokeRegistry.RemTypeRegistry Function 931
InvokeRegistry.SetRemotableDataContext Function 932
InvokeRegistry.SubstituteStrings Function 933

Structs, Records, Enums **934**
InvokeRegistry.eHeaderMethodType Enumeration 934
InvokeRegistry.ExtNameMapItem Record 935
InvokeRegistry.IntfExceptionItem Record 935
InvokeRegistry.IntfHeaderItem Record 936
InvokeRegistry.InvRegClassEntry Record 937
InvokeRegistry.InvRegIntfEntry Record 937
InvokeRegistry.MethParamNameMapItem Record 939
InvokeRegistry.ObjectConvertOptions Enumeration 939
InvokeRegistry.SerializationOptions Enumeration 940
InvokeRegistry.TDynToClear Record 941
InvokeRegistry.TIntfInvokeOption Enumeration 942
InvokeRegistry.TObjMultiOptions Enumeration 943
InvokeRegistry.TRemHeaderEntry Record 944
InvokeRegistry.TRemRegEntry Record 944

Types **946**
InvokeRegistry.ERemotableExceptionClass Type 946
InvokeRegistry.InvString Type 947
InvokeRegistry.InvStringDynArray Type 947
InvokeRegistry.PTRemotable Type 947
InvokeRegistry.TCreateInstanceProc Type 948
InvokeRegistry.TExceptionItemArray Type 948
InvokeRegistry.THeaderItemArray Type 948
InvokeRegistry.THeaderMethodTypeArray Type 949
InvokeRegistry.TIntfInvokeOptions Type 949
InvokeRegistry.TInvokableClassClass Type 949
InvokeRegistry.TInvRegIntfEntryArray Type 950
InvokeRegistry.TObjectConvertOptions Type 950
InvokeRegistry.TPascalRemotableClassRegistry Type 950
InvokeRegistry.TPascalRemotableTypeRegistry Type 951
InvokeRegistry.TRemotableClass Type 951
InvokeRegistry.TRemotableClassRegistry Type 951

InvokeRegistry.TRemotableXSClass Type 952
InvokeRegistry.TRemRegEntryArray Type 952
InvokeRegistry.TRequiredArray Type 952
InvokeRegistry.TSerializationOptions Type 953
InvokeRegistry.TSOAPAttachmentClass Type 953
InvokeRegistry.TSOAPHeaderClass Type 953

Variables **955**
InvokeRegistry.AppNameSpacePrefix Variable 955

Constants **956**
InvokeRegistry.AS_ATTRIBUTE Constant 956
InvokeRegistry.XMLBase64Types Constant 956
InvokeRegistry.XMLSchemaInstNamepspaces Constant 957
InvokeRegistry.XMLSchemaNamepspaces Constant 957

ISAPIApp Namespace 959

Classes **960**
TISAPIApplication Class 960
TISAPIApplication Members 961
TISAPIApplication.Create Constructor 961
TISAPIApplication.Destroy Destructor 961
TISAPIApplication Methods 962
TISAPIFactory Class 963
TISAPIFactory Members 963
TISAPIFactory.Create Constructor 963
TISAPIFactory Methods 964

Functions **965**
ISAPIApp.GetExtensionVersion Function 965
ISAPIApp.HttpExtensionProc Function 965
ISAPIApp.TerminateExtension Function 966

Variables **967**
ISAPIApp.DispatchThread Variable 967

IsapiHTTP Namespace 969

Classes **970**
TISAPIRequest Class 970
TISAPIRequest Members 970

TISAPIRequest.Create Constructor 971
TISAPIRequest Methods 971
TISAPIRequest Properties 976
TISAPIResponse Class 977
TISAPIResponse Members 977
TISAPIResponse.Create Constructor 978
TISAPIResponse Methods 978

Java Namespace 987

Java.Sql Namespace 988
Classes 988
SQLException Class 988

jpeg Namespace 991

Classes 992
TJPEGData Class 992
TJPEGData Members 992
TJPEGData.Destroy Destructor 993
TJPEGData Methods 993
TJPEGImage Class 994
TJPEGImage Members 994
TJPEGImage.Create Constructor 995
TJPEGImage.Destroy Destructor 997
TJPEGImage Methods 997
TJPEGImage Properties 1007

Structs, Records, Enums 1011
jpeg.TJPEGDefaults Record 1011
jpeg.TJPEGPerformance Enumeration 1012
jpeg.TJPEGPixelFormat Enumeration 1012
jpeg.TJPEGScale Enumeration 1013

Types 1014
jpeg.TJPEGQualityRange Type 1014

Variables 1015
jpeg.JPEGDefaults Variable 1015

ListActns Namespace 1017

Classes **1019**
TCustomListAction Class 1019
TCustomListAction Members 1020
TCustomListAction.Create Constructor 1020
TCustomListAction Methods 1021
TCustomListAction Properties 1023
TCustomListAction Events 1026
TCustomStaticListAction Class 1027
TCustomStaticListAction Members 1028
TCustomStaticListAction.Create Constructor 1029
TCustomStaticListAction.Destroy Destructor 1029
TCustomStaticListAction Methods 1029
TCustomStaticListAction Properties 1031
TCustomStaticListAction Events 1033
TCustomVirtualListAction Class 1034
TCustomVirtualListAction Members 1034
TCustomVirtualListAction Methods 1035
TCustomVirtualListAction Properties 1036
TCustomVirtualListAction Events 1037
TListActionLink Class 1037
TListActionLink Members 1038
TListActionLink Methods 1038
TListControlItem Class 1041
TListControlItem Members 1041
TListControlItem.Create Constructor 1042
TListControlItem Fields 1042
TListControlItem Methods 1043
TListControlItem Properties 1046
TListControlItems Class 1047
TListControlItems Members 1048
TListControlItems.Create Constructor 1049
TListControlItems Methods 1049
TListControlItems Properties 1052
TListControlItems Events 1054
TStaticListAction Class 1054
TStaticListAction Members 1055
TStaticListAction Properties 1056
TStaticListAction Events 1061
TStaticListItems Class 1063

TStaticListItems Members 1064
TStaticListItems Methods 1064
TVirtualListAction Class 1066
TVirtualListAction Members 1066
TVirtualListAction Properties 1067
TVirtualListAction Events 1071

Structs, Records, Enums **1074**
ListActns.TListItemsSortType Enumeration 1074

Types **1075**
ListActns.TGetItemCountEvent Type 1075
ListActns.TGetItemEvent Type 1075
ListActns.TGetVirtualItemEvent Type 1076
ListActns.TItemSelectedEvent Type 1076
ListActns.TListCompareEvent Type 1076
ListActns.TListControlItemClass Type 1077
ListActns.TListItemsCompare Type 1077

Mask Namespace 1079

Classes **1080**
EDBEditError Class 1080
TCustomMaskEdit Class 1080
TCustomMaskEdit Members 1081
TCustomMaskEdit.Create Constructor 1082
TCustomMaskEdit Methods 1082
TCustomMaskEdit Properties 1092
TMaskEdit Class 1100
TMaskEdit Members 1101
TMaskEdit Properties 1103
TMaskEdit Events 1141

Types **1175**
Mask.TMaskedState Type 1175

Masks Namespace 1177

Classes **1178**
EMaskException Class 1178
TMask Class 1178
TMask Members 1179

TMask.Create Constructor 1179
TMask.Destroy Destructor 1180
TMask Methods 1180

Functions 1182
Masks.MatchesMask Function 1182

MaskUtils Namespace 1183

Functions 1185
MaskUtils.FormatMaskText Function 1185
MaskUtils.IsLiteralChar Function 1187
MaskUtils.MaskDoFormatText Function 1187
MaskUtils.MaskGetCharType Function 1187
MaskUtils.MaskGetCurrentDirectives Function 1188
MaskUtils.MaskGetFldSeparator Function 1188
MaskUtils.MaskGetMaskBlank Function 1189
MaskUtils.MaskGetMaskSave Function 1189
MaskUtils.MaskIntlLiteralToChar Function 1189
MaskUtils.MaskOffsetToOffset Function 1190
MaskUtils.MaskOffsetToWideOffset Function 1190
MaskUtils.OffsetToMaskOffset Function 1191
MaskUtils.PadInputLiterals Function 1191

Structs, Records, Enums 1192
MaskUtils.TMaskCharType Enumeration 1192

Types 1193
MaskUtils.TEditMask Type 1193
MaskUtils.TMaskDirectives Type 1194
MaskUtils.TMaskedText Type 1195

Variables 1196
MaskUtils.DefaultBlank Variable 1196
MaskUtils.MaskFieldSeparator Variable 1196
MaskUtils.MaskNoSave Variable 1197

Constants 1198
MaskUtils.mDirLiteral Constant 1198
MaskUtils.mDirLowerCase Constant 1198
MaskUtils.mDirReverse Constant 1199
MaskUtils.mDirUpperCase Constant 1199

MaskUtils.mMskAlpha Constant 1200
MaskUtils.mMskAlphaNum Constant 1200
MaskUtils.mMskAlphaNumOpt Constant 1200
MaskUtils.mMskAlphaOpt Constant 1201
MaskUtils.mMskAscii Constant 1201
MaskUtils.mMskAsciiOpt Constant 1202
MaskUtils.mMskDateSeparator Constant 1202
MaskUtils.mMskNumeric Constant 1202
MaskUtils.mMskNumericOpt Constant 1203
MaskUtils.mMskNumSymOpt Constant 1203
MaskUtils.mMskTimeSeparator Constant 1204

Math Namespace 1205

Classes 1209

EInvalidArgument Class 1209

Functions 1211

Math.ArcCos Function 1213
Math.ArcCosh Function 1214
Math.ArcCot Function 1214
Math.ArcCotH Function 1215
Math.ArcCsc Function 1215
Math.ArcCscH Function 1216
Math.ArcSec Function 1216
Math.ArcSecH Function 1216
Math.ArcSin Function 1217
Math.ArcSinh Function 1217
Math.ArcTan2 Function 1218
Math.ArcTanh Function 1218
Math.Ceil Function 1219
Math.ClearExceptions Function 1219
Math.CompareValue Function 1220
Math.Cosecant Function 1220
Math.Cosh Function 1221
Math.Cot Function 1221
Math.Cotan Function 1222
Math.CotH Function 1222
Math.Csc Function 1223
Math.CscH Function 1223

Math.CycleToDeg Function 1224
Math.CycleToGrad Function 1224
Math.CycleToRad Function 1224
Math.DegToCycle Function 1225
Math.DegToGrad Function 1225
Math.DegToRad Function 1226
Math.DivMod Function 1226
Math.DoubleDecliningBalance Function 1226
Math.EnsureRange Function 1227
Math.Floor Function 1227
Math.Frexp Function 1228
Math.FutureValue Function 1228
Math.GetExceptionMask Function 1229
Math.GetPrecisionMode Function 1229
Math.GetRoundMode Function 1230
Math.GradToCycle Function 1231
Math.GradToDeg Function 1231
Math.GradToRad Function 1231
Math.Hypot Function 1232
Math.IfThen Function 1232
Math.InRange Function 1232
Math.InterestPayment Function 1233
Math.InterestRate Function 1234
Math.InternalRateOfReturn Function 1234
Math.IntPower Function 1235
Math.IsInfinite Function 1235
Math.IsNan Function 1236
Math.IsZero Function 1236
Math.Ldexp Function 1237
Math.LnXP1 Function 1237
Math.Log10 Function 1237
Math.Log2 Function 1238
Math.LogN Function 1238
Math.Max Function 1239
Math.MaxIntValue Function 1239
Math.MaxValue Function 1240
Math.Mean Function 1240
Math.MeanAndStdDev Function 1241
Math.Min Function 1241

Math.MinIntValue Function 1242
Math.MinValue Function 1242
Math.MomentSkewKurtosis Function 1243
Math.NetPresentValue Function 1243
Math.Norm Function 1244
Math.NumberOfPeriods Function 1244
Math.Payment Function 1245
Math.PeriodPayment Function 1245
Math.Poly Function 1245
Math.PopnStdDev Function 1246
Math.PopnVariance Function 1246
Math.Power Function 1247
Math.PresentValue Function 1247
Math.RadToCycle Function 1248
Math.RadToDeg Function 1248
Math.RadToGrad Function 1248
Math.RandG Function 1249
Math.RandomFrom Function 1249
Math.RandomRange Function 1250
Math.RoundTo Function 1250
Math.SameValue Function 1251
Math.Sec Function 1252
Math.Secant Function 1252
Math.SecH Function 1253
Math.SetExceptionMask Function 1253
Math.SetPrecisionMode Function 1254
Math.SetRoundMode Function 1254
Math.Sign Function 1255
Math.SimpleRoundTo Function 1255
Math.SinCos Function 1256
Math.Sinh Function 1256
Math.SLNDepreciation Function 1257
Math.StdDev Function 1257
Math.Sum Function 1258
Math.SumInt Function 1258
Math.SumOfSquares Function 1259
Math.SumsAndSquares Function 1259
Math.SYDDepreciation Function 1260
Math.Tan Function 1260

Math.Tanh Function 1261
Math.TotalVariance Function 1261
Math.Variance Function 1262

Structs, Records, Enums 1263
Math.TFPUException Enumeration 1263
Math.TFPUPrecisionMode Enumeration 1264
Math.TFPURoundingMode Enumeration 1265
Math.TPaymentTime Enumeration 1266

Types 1267
Math.TFPUExceptionMask Type 1267
Math.TRoundToEXRangeExtended Type 1268
Math.TRoundToRange Type 1268
Math.TValueSign Type 1268

Constants 1270
Math.Infinity Constant 1270
Math.MaxComp Constant 1271
Math.MaxDouble Constant 1271
Math.MaxExtended Constant 1271
Math.MaxSingle Constant 1272
Math.MinComp Constant 1272
Math.MinDouble Constant 1273
Math.MinExtended Constant 1273
Math.MinSingle Constant 1273
Math.NaN Constant 1274
Math.NegativeValue Constant 1274
Math.NegInfinity Constant 1275
Math.PositiveValue Constant 1275
Math.ZeroValue Constant 1276

MConnect Namespace 1277

Classes 1278
TCOMConnection Class 1278
TCOMConnection Members 1279
TCOMConnection Methods 1279
TCustomObjectBroker Class 1281
TCustomObjectBroker Members 1282
TCustomObjectBroker Methods 1282

TDCOMConnection Class 1284
TDCOMConnection Members 1285
TDCOMConnection.Create Constructor 1286
TDCOMConnection Methods 1286
TDCOMConnection Properties 1287
TDispatchAppServer Class 1289
TDispatchAppServer Members 1289
TDispatchAppServer.Create Constructor 1289
TDispatchAppServer Methods 1289
TDispatchConnection Class 1290
TDispatchConnection Members 1291
TDispatchConnection.Create Constructor 1291
TDispatchConnection Methods 1292
TDispatchConnection Properties 1295
TDispatchConnection Events 1302
TOLEnterpriseConnection Class 1305
TOLEnterpriseConnection Members 1306
TOLEnterpriseConnection Methods 1307
TOLEnterpriseConnection Properties 1308
TSharedConnection Class 1309
TSharedConnection Members 1310
TSharedConnection.Create Constructor 1310
TSharedConnection.Destroy Destructor 1311
TSharedConnection Methods 1311
TSharedConnection Properties 1313
TSharedConnection Events 1314

Functions **1317**
MConnect.GetMIDASAppServerList Function 1317

Menus Namespace 1319

Classes **1321**
EMenuError Class 1321
TMainMenu Class 1322
TMainMenu Members 1322
TMainMenu Methods 1323
TMainMenu Properties 1327
TMainMenu Events 1330
TMenu Class 1331

TMenu Members 1332
TMenu.Create Constructor 1333
TMenu.Destroy Destructor 1334
TMenu Methods 1334
TMenu Properties 1343
TMenu Events 1351
TMenuActionLink Class 1352
TMenuActionLink Members 1353
TMenuActionLink Fields 1353
TMenuActionLink Methods 1354
TMenuItem Class 1363
TMenuItem Members 1363
TMenuItem.Create Constructor 1366
TMenuItem.Destroy Destructor 1367
TMenuItem Methods 1368
TMenuItem Properties 1393
TMenuItem Events 1416
TMenuItemEnumerator Class 1424
TMenuItemEnumerator Members 1424
TMenuItemEnumerator.Create Constructor 1424
TMenuItemEnumerator Methods 1424
TMenuItemEnumerator Properties 1425
TMenuItemStack Class 1426
TMenuItemStack Members 1426
TMenuItemStack Methods 1426
TPopupList Class 1427
TPopupList Members 1427
TPopupList Fields 1428
TPopupList Methods 1428
TPopupList Properties 1430
TPopupMenu Class 1430
TPopupMenu Members 1431
TPopupMenu.Create Constructor 1432
TPopupMenu.Destroy Destructor 1433
TPopupMenu Methods 1433
TPopupMenu Properties 1435
TPopupMenu Events 1443

Functions **1446**

Menus.AnsiSameCaption Function 1446

Menus.DrawMenuItem Function 1447
Menus.GetHotkey Function 1447
Menus.IsAltGRPressed Function 1447
Menus.NewItem Function 1448
Menus.NewLine Function 1449
Menus.NewMenu Function 1450
Menus.NewPopupMenu Function 1451
Menus.NewSubMenu Function 1452
Menus.SameCaption Function 1453
Menus.ShortCut Function 1453
Menus.ShortCutFromMessage Function 1454
Menus.ShortCutToKey Function 1454
Menus.ShortCutToText Function 1456
Menus.StripHotkey Function 1457
Menus.TextToShortCut Function 1457

Structs, Records, Enums **1460**
Menus.TFindItemKind Enumeration 1460
Menus.TMenuAnimations Enumeration 1461
Menus.TMenuBreak Enumeration 1461
Menus.TMenuItemAutoFlag Enumeration 1462
Menus.TPopupAlignment Enumeration 1463
Menus.TTrackButton Enumeration 1463

Types **1465**
Menus.PMenuItem Type 1465
Menus.TAdvancedMenuDrawItemEvent Type 1465
Menus.TMenuActionLinkClass Type 1466
Menus.TMenuAnimation Type 1466
Menus.TMenuAutoFlag Type 1467
Menus.TMenuChangeEvent Type 1467
Menus.TMenuDrawItemEvent Type 1468
Menus.TMenuHandle Type 1468
Menus.TMenuMeasureItemEvent Type 1469

Variables **1470**
Menus.PopupList Variable 1470
Menus.ShortCutItems Variable 1470
Menus.ValidMenuHotkeys Variable 1471

Constants **1472**

Menus.cDialogSuffix Constant 1472
Menus.cHotkeyPrefix Constant 1472
Menus.cLineCaption Constant 1472

Messages Namespace 1475

Functions 1487

Messages.SendGetIntMessage Function 1487
Messages.SendGetStructMessage Function 1493
Messages.SendStructMessage Function 1499
Messages.SendTextMessage Function 1505

Structs, Records, Enums 1511

Messages.TMessage Record 1513
Messages.TMSHMouseWheel Record 1514
Messages.TWMActivate Record 1514
Messages.TWMActivateApp Record 1515
Messages.TWMAskCBFormatName Record 1515
Messages.TWMChangeCBChain Record 1516
Messages.TWMCharToItem Record 1516
Messages.TWMChooseFont_GetLogFont Record 1517
Messages.TWMCommand Record 1517
Messages.TWMCompacting Record 1518
Messages.TWMCompareItem Record 1518
Messages.TWMContextMenu Record 1519
Messages.TWMCopyData Record 1519
Messages.TWMCreate Record 1520
Messages.TWMCtlColor Record 1520
Messages.TWMDDE_Ack Record 1521
Messages.TWMDDE_Advise Record 1521
Messages.TWMDDE_Data Record 1522
Messages.TWMDDE_Execute Record 1522
Messages.TWMDDE_Initiate Record 1523
Messages.TWMDDE_Poke Record 1523
Messages.TWMDDE_Request Record 1524
Messages.TWMDDE_Terminate Record 1524
Messages.TWMDDE_Unadvise Record 1525
Messages.TWMDeleteItem Record 1525
Messages.TWMDevModeChange Record 1526
Messages.TWMDisplayChange Record 1526

Messages.TWMDrawItem Record 1527
Messages.TWMDropFiles Record 1527
Messages.TWMEnable Record 1528
Messages.TWMEndSession Record 1528
Messages.TWMEnterIdle Record 1529
Messages.TWMEnterMenuLoop Record 1529
Messages.TWMEraseBkgnd Record 1530
Messages.TWMGetIcon Record 1530
Messages.TWMGetMinMaxInfo Record 1531
Messages.TWMGetText Record 1531
Messages.TWMHelp Record 1532
Messages.TWMHotKey Record 1532
Messages.TWMHScrollClipboard Record 1533
Messages.TWMInitDialog Record 1533
Messages.TWMInitMenu Record 1534
Messages.TWMInitMenuPopup Record 1534
Messages.TWMKey Record 1535
Messages.TWMKillFocus Record 1536
Messages.TWMMDIActivate Record 1536
Messages.TWMMDICascade Record 1537
Messages.TWMMDICreate Record 1537
Messages.TWMMDIDestroy Record 1538
Messages.TWMMDIMaximize Record 1538
Messages.TWMMDINext Record 1539
Messages.TWMMDIRestore Record 1539
Messages.TWMMDISetMenu Record 1540
Messages.TWMMDITile Record 1540
Messages.TWMMeasureItem Record 1541
Messages.TWMMenuChar Record 1541
Messages.TWMMenuSelect Record 1542
Messages.TWMMouse Record 1543
Messages.TWMMouseActivate Record 1544
Messages.TWMMouseWheel Record 1544
Messages.TWMMove Record 1545
Messages.TWMMoving Record 1545
Messages.TWMNCActivate Record 1546
Messages.TWMNCCalcSize Record 1546
Messages.TWMNCCreate Record 1547
Messages.TWMNCHitMessage Record 1547

Messages.TWMNCHitTest Record 1548
Messages.TWMNCPaint Record 1548
Messages.TWMNextDlgCtl Record 1549
Messages.TWMNoParams Record 1549
Messages.TWMNotify Record 1550
Messages.TWMNotifyFormat Record 1550
Messages.TWMPaint Record 1551
Messages.TWMPaintClipboard Record 1551
Messages.TWMPaletteChanged Record 1552
Messages.TWMPaletteIsChanging Record 1552
Messages.TWMParentNotify Record 1553
Messages.TWMPower Record 1554
Messages.TWMPrint Record 1554
Messages.TWMQueryEndSession Record 1555
Messages.TWMQuit Record 1555
Messages.TWMRenderFormat Record 1556
Messages.TWMScroll Record 1556
Messages.TWMSetCursor Record 1557
Messages.TWMSetFocus Record 1557
Messages.TWMSetFont Record 1558
Messages.TWMSetHotKey Record 1558
Messages.TWMSetIcon Record 1559
Messages.TWMSetRedraw Record 1559
Messages.TWMSetText Record 1560
Messages.TWMSettingChange Record 1560
Messages.TWMShowWindow Record 1561
Messages.TWMSize Record 1561
Messages.TWMSizeClipboard Record 1562
Messages.TWMSpoolerStatus Record 1562
Messages.TWMStyleChange Record 1563
Messages.TWMSysCommand Record 1563
Messages.TWMSysDeadChar Record 1564
Messages.TWMSystemError Record 1564
Messages.TWMTimer Record 1565
Messages.TWMUIState Record 1565
Messages.TWMVScrollClipboard Record 1566
Messages.TWMWindowPosMsg Record 1566
Messages.TWMWinIniChange Record 1567

Types **1568**

Messages.PMessage Type 1569
Messages.TWMCancelMode Type 1570
Messages.TWMChangeUIState Type 1570
Messages.TWMChar Type 1570
Messages.TWMChildActivate Type 1571
Messages.TWMClear Type 1571
Messages.TWMClose Type 1571
Messages.TWMCopy Type 1572
Messages.TWMCtlColorBtn Type 1572
Messages.TWMCtlColorDlg Type 1572
Messages.TWMCtlColorEdit Type 1573
Messages.TWMCtlColorListbox Type 1573
Messages.TWMCtlColorMsgbox Type 1573
Messages.TWMCtlColorScrollbar Type 1574
Messages.TWMCtlColorStatic Type 1574
Messages.TWMCut Type 1574
Messages.TWMDeadChar Type 1575
Messages.TWMDestroy Type 1575
Messages.TWMDestroyClipboard Type 1575
Messages.TWMDrawClipboard Type 1576
Messages.TWMExitMenuLoop Type 1576
Messages.TWMFontChange Type 1576
Messages.TWMGetDlgCode Type 1577
Messages.TWMGetFont Type 1577
Messages.TWMGetHotKey Type 1577
Messages.TWMGetTextLength Type 1578
Messages.TWMHScroll Type 1578
Messages.TWMIconEraseBkgnd Type 1578
Messages.TWMKeyDown Type 1579
Messages.TWMKeyUp Type 1579
Messages.TWMLButtonDblClk Type 1579
Messages.TWMLButtonDown Type 1580
Messages.TWMLButtonUp Type 1580
Messages.TWMMButtonDblClk Type 1580
Messages.TWMMButtonDown Type 1581
Messages.TWMMButtonUp Type 1581
Messages.TWMMDIGetActive Type 1581
Messages.TWMMDIIconArrange Type 1582
Messages.TWMMDIRefreshMenu Type 1582

Messages.TWMMouseMove Type 1582
Messages.TWMNCDestroy Type 1583
Messages.TWMNCLButtonDblClk Type 1583
Messages.TWMNCLButtonDown Type 1583
Messages.TWMNCLButtonUp Type 1584
Messages.TWMNCMButtonDblClk Type 1584
Messages.TWMNCMButtonDown Type 1584
Messages.TWMNCMButtonUp Type 1585
Messages.TWMNCMouseMove Type 1585
Messages.TWMNCRButtonDblClk Type 1585
Messages.TWMNCRButtonDown Type 1586
Messages.TWMNCRButtonUp Type 1586
Messages.TWMPaintIcon Type 1586
Messages.TWMPaste Type 1587
Messages.TWMPrintClient Type 1587
Messages.TWMQueryDragIcon Type 1587
Messages.TWMQueryNewPalette Type 1588
Messages.TWMQueryOpen Type 1588
Messages.TWMQueryUIState Type 1588
Messages.TWMQueueSync Type 1589
Messages.TWMRButtonDblClk Type 1589
Messages.TWMRButtonDown Type 1589
Messages.TWMRButtonUp Type 1590
Messages.TWMRenderAllFormats Type 1590
Messages.TWMStyleChanged Type 1590
Messages.TWMStyleChanging Type 1591
Messages.TWMSysChar Type 1591
Messages.TWMSysColorChange Type 1591
Messages.TWMSysKeyDown Type 1592
Messages.TWMSysKeyUp Type 1592
Messages.TWMTimeChange Type 1592
Messages.TWMUndo Type 1593
Messages.TWMUpdateUIState Type 1593
Messages.TWMVKeyToItem Type 1593
Messages.TWMVScroll Type 1594
Messages.TWMWindowPosChanged Type 1594
Messages.TWMWindowPosChanging Type 1594

Constants **1595**

Messages.BM_CLICK Constant 1602

Messages.BM_GETCHECK Constant 1603
Messages.BM_GETIMAGE Constant 1603
Messages.BM_GETSTATE Constant 1604
Messages.BM_SETCHECK Constant 1604
Messages.BM_SETIMAGE Constant 1604
Messages.BM_SETSTATE Constant 1605
Messages.BM_SETSTYLE Constant 1605
Messages.BN_CLICKED Constant 1606
Messages.BN_DBLCLK Constant 1606
Messages.BN_DISABLE Constant 1607
Messages.BN_DOUBLECLICKED Constant 1607
Messages.BN_HILITE Constant 1607
Messages.BN_KILLFOCUS Constant 1608
Messages.BN_PAINT Constant 1608
Messages.BN_PUSHED Constant 1609
Messages.BN_SETFOCUS Constant 1609
Messages.BN_UNHILITE Constant 1610
Messages.BN_UNPUSHED Constant 1610
Messages.CB_ADDSTRING Constant 1610
Messages.CB_DELETESTRING Constant 1611
Messages.CB_DIR Constant 1611
Messages.CB_FINDSTRING Constant 1612
Messages.CB_FINDSTRINGEXACT Constant 1612
Messages.CB_GETCOUNT Constant 1613
Messages.CB_GETCURSEL Constant 1613
Messages.CB_GETDROPPEDCONTROLRECT Constant 1613
Messages.CB_GETDROPPEDSTATE Constant 1614
Messages.CB_GETDROPPEDWIDTH Constant 1614
Messages.CB_GETEDITSEL Constant 1615
Messages.CB_GETEXTENDEDUI Constant 1615
Messages.CB_GETHORIZONTALEXTENT Constant 1616
Messages.CB_GETITEMDATA Constant 1616
Messages.CB_GETITEMHEIGHT Constant 1617
Messages.CB_GETLBTEXT Constant 1617
Messages.CB_GETLBTEXTLEN Constant 1617
Messages.CB_GETLOCALE Constant 1618
Messages.CB_GETTOPINDEX Constant 1618
Messages.CB_INITSTORAGE Constant 1619
Messages.CB_INSERTSTRING Constant 1619

Messages.CB_LIMITTEXT Constant 1620
Messages.CB_MSGMAX Constant 1620
Messages.CB_RESETCONTENT Constant 1620
Messages.CB_SELECTSTRING Constant 1621
Messages.CB_SETCURSEL Constant 1621
Messages.CB_SETDROPPEDWIDTH Constant 1622
Messages.CB_SETEDITSEL Constant 1622
Messages.CB_SETEXTENDEDUI Constant 1623
Messages.CB_SETHORIZONTALEXTENT Constant 1623
Messages.CB_SETITEMDATA Constant 1623
Messages.CB_SETITEMHEIGHT Constant 1624
Messages.CB_SETLOCALE Constant 1624
Messages.CB_SETTOPINDEX Constant 1625
Messages.CB_SHOWDROPDOWN Constant 1625
Messages.CBN_CLOSEUP Constant 1626
Messages.CBN_DBLCLK Constant 1626
Messages.CBN_DROPDOWN Constant 1626
Messages.CBN_EDITCHANGE Constant 1627
Messages.CBN_EDITUPDATE Constant 1627
Messages.CBN_ERRSPACE Constant 1628
Messages.CBN_KILLFOCUS Constant 1628
Messages.CBN_SELCHANGE Constant 1629
Messages.CBN_SELENDCANCEL Constant 1629
Messages.CBN_SELENDOK Constant 1629
Messages.CBN_SETFOCUS Constant 1630
Messages.DM_GETDEFID Constant 1630
Messages.DM_REPOSITION Constant 1631
Messages.DM_SETDEFID Constant 1631
Messages.EM_CANUNDO Constant 1632
Messages.EM_CHARFROMPOS Constant 1632
Messages.EM_EMPTYUNDOBUFFER Constant 1632
Messages.EM_FMTLINES Constant 1633
Messages.EM_GETFIRSTVISIBLELINE Constant 1633
Messages.EM_GETHANDLE Constant 1634
Messages.EM_GETIMESTATUS Constant 1634
Messages.EM_GETLIMITTEXT Constant 1635
Messages.EM_GETLINE Constant 1635
Messages.EM_GETLINECOUNT Constant 1635
Messages.EM_GETMARGINS Constant 1636

Messages.EM_GETMODIFY Constant 1636
Messages.EM_GETPASSWORDCHAR Constant 1637
Messages.EM_GETRECT Constant 1637
Messages.EM_GETSEL Constant 1638
Messages.EM_GETTHUMB Constant 1638
Messages.EM_GETWORDBREAKPROC Constant 1638
Messages.EM_LIMITTEXT Constant 1639
Messages.EM_LINEFROMCHAR Constant 1639
Messages.EM_LINEINDEX Constant 1640
Messages.EM_LINELENGTH Constant 1640
Messages.EM_LINESCROLL Constant 1641
Messages.EM_POSFROMCHAR Constant 1641
Messages.EM_REPLACESEL Constant 1641
Messages.EM_SCROLL Constant 1642
Messages.EM_SCROLLCARET Constant 1642
Messages.EM_SETHANDLE Constant 1643
Messages.EM_SETIMESTATUS Constant 1643
Messages.EM_SETLIMITTEXT Constant 1644
Messages.EM_SETMARGINS Constant 1644
Messages.EM_SETMODIFY Constant 1644
Messages.EM_SETPASSWORDCHAR Constant 1645
Messages.EM_SETREADONLY Constant 1645
Messages.EM_SETRECT Constant 1646
Messages.EM_SETRECTNP Constant 1646
Messages.EM_SETSEL Constant 1647
Messages.EM_SETTABSTOPS Constant 1647
Messages.EM_SETWORDBREAKPROC Constant 1647
Messages.EM_UNDO Constant 1648
Messages.EN_CHANGE Constant 1648
Messages.EN_ERRSPACE Constant 1649
Messages.EN_HSCROLL Constant 1649
Messages.EN_KILLFOCUS Constant 1650
Messages.EN_MAXTEXT Constant 1650
Messages.EN_SETFOCUS Constant 1650
Messages.EN_UPDATE Constant 1651
Messages.EN_VSCROLL Constant 1651
Messages.LB_ADDFILE Constant 1652
Messages.LB_ADDSTRING Constant 1652
Messages.LB_DELETESTRING Constant 1653

Messages.LB_DIR Constant 1653
Messages.LB_FINDSTRING Constant 1653
Messages.LB_FINDSTRINGEXACT Constant 1654
Messages.LB_GETANCHORINDEX Constant 1654
Messages.LB_GETCARETINDEX Constant 1655
Messages.LB_GETCOUNT Constant 1655
Messages.LB_GETCURSEL Constant 1656
Messages.LB_GETHORIZONTALEXTENT Constant 1656
Messages.LB_GETITEMDATA Constant 1656
Messages.LB_GETITEMHEIGHT Constant 1657
Messages.LB_GETITEMRECT Constant 1657
Messages.LB_GETLOCALE Constant 1658
Messages.LB_GETSEL Constant 1658
Messages.LB_GETSELCOUNT Constant 1659
Messages.LB_GETSELITEMS Constant 1659
Messages.LB_GETTEXT Constant 1659
Messages.LB_GETTEXTLEN Constant 1660
Messages.LB_GETTOPINDEX Constant 1660
Messages.LB_INITSTORAGE Constant 1661
Messages.LB_INSERTSTRING Constant 1661
Messages.LB_ITEMFROMPOINT Constant 1662
Messages.LB_MSGMAX Constant 1662
Messages.LB_RESETCONTENT Constant 1662
Messages.LB_SELECTSTRING Constant 1663
Messages.LB_SELITEMRANGE Constant 1663
Messages.LB_SELITEMRANGEEX Constant 1664
Messages.LB_SETANCHORINDEX Constant 1664
Messages.LB_SETCARETINDEX Constant 1665
Messages.LB_SETCOLUMNWIDTH Constant 1665
Messages.LB_SETCOUNT Constant 1665
Messages.LB_SETCURSEL Constant 1666
Messages.LB_SETHORIZONTALEXTENT Constant 1666
Messages.LB_SETITEMDATA Constant 1667
Messages.LB_SETITEMHEIGHT Constant 1667
Messages.LB_SETLOCALE Constant 1668
Messages.LB_SETSEL Constant 1668
Messages.LB_SETTABSTOPS Constant 1668
Messages.LB_SETTOPINDEX Constant 1669
Messages.LBN_DBLCLK Constant 1669

Messages.LBN_ERRSPACE Constant 1670
Messages.LBN_KILLFOCUS Constant 1670
Messages.LBN_SELCANCEL Constant 1671
Messages.LBN_SELCHANGE Constant 1671
Messages.LBN_SETFOCUS Constant 1671
Messages.PSM_PAGEINFO Constant 1672
Messages.PSM_SHEETINFO Constant 1672
Messages.SBM_ENABLE_ARROWS Constant 1673
Messages.SBM_GETPOS Constant 1673
Messages.SBM_GETRANGE Constant 1674
Messages.SBM_GETSCROLLBARINFO Constant 1674
Messages.SBM_GETSCROLLINFO Constant 1674
Messages.SBM_SETPOS Constant 1675
Messages.SBM_SETRANGE Constant 1675
Messages.SBM_SETRANGEREDRAW Constant 1676
Messages.SBM_SETSCROLLINFO Constant 1676
Messages.WM_ACTIVATE Constant 1677
Messages.WM_ACTIVATEAPP Constant 1677
Messages.WM_APP Constant 1678
Messages.WM_APPCOMMAND Constant 1678
Messages.WM_ASKCBFORMATNAME Constant 1678
Messages.WM_CANCELJOURNAL Constant 1679
Messages.WM_CANCELMODE Constant 1679
Messages.WM_CAPTURECHANGED Constant 1680
Messages.WM_CHANGECBCHAIN Constant 1680
Messages.WM_CHANGEUISTATE Constant 1681
Messages.WM_CHAR Constant 1681
Messages.WM_CHARTOITEM Constant 1681
Messages.WM_CHILDACTIVATE Constant 1682
Messages.WM_CLEAR Constant 1682
Messages.WM_CLOSE Constant 1683
Messages.WM_COALESCE_FIRST Constant 1683
Messages.WM_COALESCE_LAST Constant 1684
Messages.WM_COMMAND Constant 1684
Messages.WM_COMMNOTIFY Constant 1685
Messages.WM_COMPACTING Constant 1685
Messages.WM_COMPAREITEM Constant 1685
Messages.WM_CONTEXTMENU Constant 1686
Messages.WM_COPY Constant 1686

Messages.WM_COPYDATA Constant 1687
Messages.WM_CREATE Constant 1687
Messages.WM_CTLCOLOR Constant 1688
Messages.WM_CTLCOLORBTN Constant 1688
Messages.WM_CTLCOLORDLG Constant 1688
Messages.WM_CTLCOLOREDIT Constant 1689
Messages.WM_CTLCOLORLISTBOX Constant 1689
Messages.WM_CTLCOLORMSGBOX Constant 1690
Messages.WM_CTLCOLORSCROLLBAR Constant 1690
Messages.WM_CTLCOLORSTATIC Constant 1691
Messages.WM_CUT Constant 1691
Messages.WM_DDE_ACK Constant 1691
Messages.WM_DDE_ADVISE Constant 1692
Messages.WM_DDE_DATA Constant 1692
Messages.WM_DDE_EXECUTE Constant 1693
Messages.WM_DDE_FIRST Constant 1693
Messages.WM_DDE_INITIATE Constant 1694
Messages.WM_DDE_LAST Constant 1694
Messages.WM_DDE_POKE Constant 1695
Messages.WM_DDE_REQUEST Constant 1695
Messages.WM_DDE_TERMINATE Constant 1695
Messages.WM_DDE_UNADVISE Constant 1696
Messages.WM_DEADCHAR Constant 1696
Messages.WM_DELETEITEM Constant 1697
Messages.WM_DESTROY Constant 1697
Messages.WM_DESTROYCLIPBOARD Constant 1698
Messages.WM_DEVICECHANGE Constant 1698
Messages.WM_DEVMODECHANGE Constant 1698
Messages.WM_DISPLAYCHANGE Constant 1699
Messages.WM_DRAWCLIPBOARD Constant 1699
Messages.WM_DRAWITEM Constant 1700
Messages.WM_DROPFILES Constant 1700
Messages.WM_DWMCOLORIZATIONCOLORCHANGED Constant 1701
Messages.WM_DWMCOMPOSITIONCHANGED Constant 1701
Messages.WM_DWMNCRENDERINGCHANGED Constant 1702
Messages.WM_DWMWINDOWMAXIMIZEDCHANGE Constant 1702
Messages.WM_ENABLE Constant 1703
Messages.WM_ENDSESSION Constant 1703
Messages.WM_ENTERIDLE Constant 1703

Messages.WM_ENTERMENULOOP Constant 1704
Messages.WM_ENTERSIZEMOVE Constant 1704
Messages.WM_ERASEBKGND Constant 1705
Messages.WM_EXITMENULOOP Constant 1705
Messages.WM_EXITSIZEMOVE Constant 1706
Messages.WM_FONTCHANGE Constant 1706
Messages.WM_GETDLGCODE Constant 1707
Messages.WM_GETFONT Constant 1707
Messages.WM_GETHOTKEY Constant 1707
Messages.WM_GETICON Constant 1708
Messages.WM_GETMINMAXINFO Constant 1708
Messages.WM_GETOBJECT Constant 1709
Messages.WM_GETTEXT Constant 1709
Messages.WM_GETTEXTLENGTH Constant 1710
Messages.WM_HANDHELDFIRST Constant 1710
Messages.WM_HANDHELDLAST Constant 1711
Messages.WM_HELP Constant 1711
Messages.WM_HOTKEY Constant 1711
Messages.WM_HSCROLL Constant 1712
Messages.WM_HSCROLLCLIPBOARD Constant 1712
Messages.WM_ICONERASEBKGND Constant 1713
Messages.WM_IME_CHAR Constant 1713
Messages.WM_IME_COMPOSITION Constant 1714
Messages.WM_IME_COMPOSITIONFULL Constant 1714
Messages.WM_IME_CONTROL Constant 1715
Messages.WM_IME_ENDCOMPOSITION Constant 1715
Messages.WM_IME_KEYDOWN Constant 1715
Messages.WM_IME_KEYLAST Constant 1716
Messages.WM_IME_KEYUP Constant 1716
Messages.WM_IME_NOTIFY Constant 1717
Messages.WM_IME_REQUEST Constant 1717
Messages.WM_IME_SELECT Constant 1718
Messages.WM_IME_SETCONTEXT Constant 1718
Messages.WM_IME_STARTCOMPOSITION Constant 1718
Messages.WM_INITDIALOG Constant 1719
Messages.WM_INITMENU Constant 1719
Messages.WM_INITMENUPOPUP Constant 1720
Messages.WM_INPUT Constant 1720
Messages.WM_INPUTLANGCHANGE Constant 1721

Messages.WM_INPUTLANGCHANGEREQUEST Constant 1721
Messages.WM_KEYDOWN Constant 1722
Messages.WM_KEYFIRST Constant 1722
Messages.WM_KEYLAST Constant 1722
Messages.WM_KEYUP Constant 1723
Messages.WM_KILLFOCUS Constant 1723
Messages.WM_LBUTTONDBLCLK Constant 1724
Messages.WM_LBUTTONDOWN Constant 1724
Messages.WM_LBUTTONUP Constant 1725
Messages.WM_MBUTTONDBLCLK Constant 1725
Messages.WM_MBUTTONDOWN Constant 1726
Messages.WM_MBUTTONUP Constant 1726
Messages.WM_MDIACTIVATE Constant 1726
Messages.WM_MDICASCADE Constant 1727
Messages.WM_MDICREATE Constant 1727
Messages.WM_MDIDESTROY Constant 1728
Messages.WM_MDIGETACTIVE Constant 1728
Messages.WM_MDIICONARRANGE Constant 1729
Messages.WM_MDIMAXIMIZE Constant 1729
Messages.WM_MDINEXT Constant 1729
Messages.WM_MDIREFRESHMENU Constant 1730
Messages.WM_MDIRESTORE Constant 1730
Messages.WM_MDISETMENU Constant 1731
Messages.WM_MDITILE Constant 1731
Messages.WM_MEASUREITEM Constant 1732
Messages.WM_MENUCHAR Constant 1732
Messages.WM_MENUCOMMAND Constant 1732
Messages.WM_MENUDRAG Constant 1733
Messages.WM_MENUGETOBJECT Constant 1733
Messages.WM_MENURBUTTONUP Constant 1734
Messages.WM_MENUSELECT Constant 1734
Messages.WM_MOUSEACTIVATE Constant 1735
Messages.WM_MOUSEFIRST Constant 1735
Messages.WM_MOUSEHOVER Constant 1736
Messages.WM_MOUSELAST Constant 1736
Messages.WM_MOUSELEAVE Constant 1736
Messages.WM_MOUSEMOVE Constant 1737
Messages.WM_MOUSEWHEEL Constant 1737
Messages.WM_MOVE Constant 1738

Messages.WM_MOVING Constant 1738
Messages.WM_NCACTIVATE Constant 1739
Messages.WM_NCCALCSIZE Constant 1739
Messages.WM_NCCREATE Constant 1739
Messages.WM_NCDESTROY Constant 1740
Messages.WM_NCHITTEST Constant 1740
Messages.WM_NCLBUTTONDBLCLK Constant 1741
Messages.WM_NCLBUTTONDOWN Constant 1741
Messages.WM_NCLBUTTONUP Constant 1742
Messages.WM_NCMBUTTONDBLCLK Constant 1742
Messages.WM_NCMBUTTONDOWN Constant 1742
Messages.WM_NCMBUTTONUP Constant 1743
Messages.WM_NCMOUSEHOVER Constant 1743
Messages.WM_NCMOUSELEAVE Constant 1744
Messages.WM_NCMOUSEMOVE Constant 1744
Messages.WM_NCPAINT Constant 1745
Messages.WM_NCRBUTTONDBLCLK Constant 1745
Messages.WM_NCRBUTTONDOWN Constant 1746
Messages.WM_NCRBUTTONUP Constant 1746
Messages.WM_NCXBUTTONDBLCLK Constant 1746
Messages.WM_NCXBUTTONDOWN Constant 1747
Messages.WM_NCXBUTTONUP Constant 1747
Messages.WM_NEXTDLGCTL Constant 1748
Messages.WM_NEXTMENU Constant 1748
Messages.WM_NOTIFY Constant 1749
Messages.WM_NOTIFYFORMAT Constant 1749
Messages.WM_NULL Constant 1749
Messages.WM_PAINT Constant 1750
Messages.WM_PAINTCLIPBOARD Constant 1750
Messages.WM_PAINTICON Constant 1751
Messages.WM_PALETTECHANGED Constant 1751
Messages.WM_PALETTEISCHANGING Constant 1752
Messages.WM_PARENTNOTIFY Constant 1752
Messages.WM_PASTE Constant 1752
Messages.WM_PENWINFIRST Constant 1753
Messages.WM_PENWINLAST Constant 1753
Messages.WM_POWER Constant 1754
Messages.WM_POWERBROADCAST Constant 1754
Messages.WM_PRINT Constant 1755

Messages.WM_PRINTCLIENT Constant 1755
Messages.WM_QUERYDRAGICON Constant 1756
Messages.WM_QUERYENDSESSION Constant 1756
Messages.WM_QUERYNEWPALETTE Constant 1756
Messages.WM_QUERYOPEN Constant 1757
Messages.WM_QUERYUISTATE Constant 1757
Messages.WM_QUEUESYNC Constant 1758
Messages.WM_QUIT Constant 1758
Messages.WM_RBUTTONDBLCLK Constant 1759
Messages.WM_RBUTTONDOWN Constant 1759
Messages.WM_RBUTTONUP Constant 1759
Messages.WM_RENDERALLFORMATS Constant 1760
Messages.WM_RENDERFORMAT Constant 1760
Messages.WM_SETCURSOR Constant 1761
Messages.WM_SETFOCUS Constant 1761
Messages.WM_SETFONT Constant 1762
Messages.WM_SETHOTKEY Constant 1762
Messages.WM_SETICON Constant 1763
Messages.WM_SETREDRAW Constant 1763
Messages.WM_SETTEXT Constant 1763
Messages.WM_SETTINGCHANGE Constant 1764
Messages.WM_SHOWWINDOW Constant 1764
Messages.WM_SIZE Constant 1765
Messages.WM_SIZECLIPBOARD Constant 1765
Messages.WM_SIZING Constant 1766
Messages.WM_SPOOLERSTATUS Constant 1766
Messages.WM_STYLECHANGED Constant 1766
Messages.WM_STYLECHANGING Constant 1767
Messages.WM_SYSCHAR Constant 1767
Messages.WM_SYSCOLORCHANGE Constant 1768
Messages.WM_SYSCOMMAND Constant 1768
Messages.WM_SYSDEADCHAR Constant 1769
Messages.WM_SYSKEYDOWN Constant 1769
Messages.WM_SYSKEYUP Constant 1770
Messages.WM_SYSTEMERROR Constant 1770
Messages.WM_TABLET_FIRST Constant 1770
Messages.WM_TABLET_LAST Constant 1771
Messages.WM_TCARD Constant 1771
Messages.WM_THEMECHANGED Constant 1772

Messages.WM_TIMECHANGE Constant 1772
Messages.WM_TIMER Constant 1773
Messages.WM_UNDO Constant 1773
Messages.WM_UNICHAR Constant 1774
Messages.WM_UNINITMENUPOPUP Constant 1774
Messages.WM_UPDATEUISTATE Constant 1774
Messages.WM_USER Constant 1775
Messages.WM_USERCHANGED Constant 1775
Messages.WM_VKEYTOITEM Constant 1776
Messages.WM_VSCROLL Constant 1776
Messages.WM_VSCROLLCLIPBOARD Constant 1777
Messages.WM_WINDOWPOSCHANGED Constant 1777
Messages.WM_WINDOWPOSCHANGING Constant 1778
Messages.WM_WININICHANGE Constant 1778
Messages.WM_WTSSESSION_CHANGE Constant 1778

MidasCon Namespace 1781

Classes **1782**
TMidasConnection Class 1782
TMidasConnection Members 1782
TMidasConnection.Create Constructor 1783
TMidasConnection Methods 1784
TMidasConnection Properties 1785
TRemoteServer Class 1787

Structs, Records, Enums **1789**
MidasCon.TConnectType Enumeration 1789

MidItems Namespace 1791

Classes **1796**
TApplyUpdatesButton Class 1798
TApplyUpdatesButton Members 1799
TApplyUpdatesButton.Create Constructor 1799
TApplyUpdatesButton Methods 1800
TApplyUpdatesButton Properties 1801
TCaptionAttributes Class 1804
TCaptionAttributes Members 1804
TCaptionAttributes.Create Constructor 1805

TCaptionAttributes Properties 1805
TCustomDataForm Class 1808
TCustomDataForm Members 1808
TCustomDataForm Methods 1809
TCustomDataGrid Class 1811
TCustomDataGrid Members 1812
TCustomDataGrid.Create Constructor 1814
TCustomDataGrid.Destroy Destructor 1814
TCustomDataGrid Methods 1814
TCustomDataGrid Properties 1817
TCustomDataNavigator Class 1819
TCustomDataNavigator Members 1820
TCustomDataNavigator Methods 1821
TCustomDataNavigator Properties 1823
TCustomFieldGroup Class 1824
TCustomFieldGroup Members 1825
TCustomFieldGroup.Create Constructor 1827
TCustomFieldGroup.Destroy Destructor 1827
TCustomFieldGroup Methods 1828
TCustomFieldGroup Properties 1829
TCustomLayoutGroup Class 1830
TCustomLayoutGroup Members 1831
TCustomLayoutGroup.Create Constructor 1832
TCustomLayoutGroup.Destroy Destructor 1832
TCustomLayoutGroup Methods 1832
TCustomLayoutGroup Properties 1834
TCustomQueryButtons Class 1836
TCustomQueryButtons Members 1836
TCustomQueryButtons Methods 1837
TCustomQueryButtons Properties 1839
TCustomQueryForm Class 1841
TCustomQueryForm Members 1842
TCustomQueryForm.Create Constructor 1843
TCustomQueryForm Methods 1843
TCustomQueryForm Properties 1846
TDataForm Class 1847
TDataForm Members 1848
TDataForm Properties 1849
TDataGrid Class 1850

TDataGrid Members 1851
TDataGrid Properties 1853
TDataNavigator Class 1855
TDataNavigator Members 1856
TDataNavigator Properties 1857
TDataSetButton Class 1859
TDataSetButton Members 1860
TDataSetButton Fields 1861
TDataSetButton Methods 1862
TDataSetButton Properties 1864
TDeleteButton Class 1867
TDeleteButton Members 1867
TDeleteButton.Create Constructor 1869
TFieldGroup Class 1869
TFieldGroup Members 1870
TFieldGroup Methods 1872
TFieldGroup Properties 1873
TFieldRadioGroup Class 1874
TFieldRadioGroup Members 1875
TFieldRadioGroup Properties 1877
TFieldSelectOptions Class 1884
TFieldSelectOptions Members 1885
TFieldSelectOptions Properties 1887
TFieldStatus Class 1894
TFieldStatus Members 1894
TFieldStatus Properties 1896
TFieldText Class 1899
TFieldText Members 1899
TFieldText Properties 1901
TFieldTextArea Class 1905
TFieldTextArea Members 1906
TFieldTextArea Properties 1908
TFirstButton Class 1913
TFirstButton Members 1913
TFirstButton.Create Constructor 1915
TFormLayout Class 1915
TFormLayout Members 1916
TFormLayout.Create Constructor 1916
TFormLayout Methods 1917

TFormLayout Properties 1921
TGridAttributes Class 1923
TGridAttributes Members 1923
TGridAttributes.Create Constructor 1924
TGridAttributes Methods 1924
TGridAttributes Properties 1925
TGridRowAttributes Class 1929
TGridRowAttributes Members 1930
TGridRowAttributes.Create Constructor 1930
TGridRowAttributes Methods 1931
TGridRowAttributes Properties 1932
TInsertButton Class 1935
TInsertButton Members 1936
TInsertButton.Create Constructor 1937
TLastButton Class 1937
TLastButton Members 1938
TLastButton.Create Constructor 1939
TLayoutGroup Class 1939
TLayoutGroup Members 1940
TLayoutGroup Properties 1941
TNextButton Class 1943
TNextButton Members 1943
TNextButton.Create Constructor 1944
TNextPageButton Class 1945
TNextPageButton Members 1945
TNextPageButton.Create Constructor 1947
TPostButton Class 1947
TPostButton Members 1948
TPostButton.Create Constructor 1949
TPriorButton Class 1949
TPriorButton Members 1950
TPriorButton.Create Constructor 1951
TPriorPageButton Class 1951
TPriorPageButton Members 1952
TPriorPageButton.Create Constructor 1953
TQueryButton Class 1953
TQueryButton Members 1954
TQueryButton.Create Constructor 1955
TQueryButton Fields 1955

TQueryButton Methods 1956
TQueryButton Properties 1959
TQueryButtons Class 1962
TQueryButtons Members 1963
TQueryButtons Properties 1964
TQueryFieldGroup Class 1965
TQueryFieldGroup Members 1967
TQueryFieldGroup Methods 1969
TQueryFieldGroup Properties 1970
TQueryForm Class 1971
TQueryForm Members 1972
TQueryForm Properties 1973
TQueryRadioGroup Class 1976
TQueryRadioGroup Members 1977
TQueryRadioGroup Methods 1979
TQueryRadioGroup Properties 1980
TQuerySelectOptions Class 1989
TQuerySelectOptions Members 1989
TQuerySelectOptions Methods 1992
TQuerySelectOptions Properties 1993
TQueryText Class 2000
TQueryText Members 2001
TQueryText.Create Constructor 2003
TQueryText Methods 2003
TQueryText Properties 2005
TQueryTextArea Class 2010
TQueryTextArea Members 2011
TQueryTextArea.Create Constructor 2013
TQueryTextArea.Destroy Destructor 2013
TQueryTextArea Methods 2013
TQueryTextArea Properties 2015
TResetQueryButton Class 2022
TResetQueryButton Members 2022
TResetQueryButton.Create Constructor 2023
TResetQueryButton Methods 2023
TSelectOptionsColumn Class 2024
TSelectOptionsColumn Members 2025
TSelectOptionsColumn Methods 2027
TSelectOptionsColumn Properties 2027

TStatusColumn Class 2033
TStatusColumn Members 2034
TStatusColumn Methods 2035
TStatusColumn Properties 2036
TSubmitQueryButton Class 2038
TSubmitQueryButton Members 2039
TSubmitQueryButton.Create Constructor 2040
TSubmitQueryButton Methods 2040
TTextAreaColumn Class 2041
TTextAreaColumn Members 2041
TTextAreaColumn Methods 2043
TTextAreaColumn Properties 2044
TTextColumn Class 2049
TTextColumn Members 2049
TTextColumn Methods 2051
TTextColumn Properties 2051
TUndoButton Class 2055
TUndoButton Members 2056
TUndoButton.Create Constructor 2057
TWebButton Class 2058
TWebButton Members 2059
TWebButton.Create Constructor 2059
TWebButton Methods 2059
TWebButton Properties 2062
TWebControlGroup Class 2064
TWebControlGroup Members 2065
TWebControlGroup.Create Constructor 2065
TWebControlGroup.Destroy Destructor 2066
TWebControlGroup Methods 2066
TWebControlGroup Properties 2072
TWebDataDisplay Class 2073
TWebDataDisplay Members 2074
TWebDataDisplay.Create Constructor 2075
TWebDataDisplay.Destroy Destructor 2075
TWebDataDisplay Methods 2075
TWebDataDisplay Properties 2083
TWebDataInput Class 2087
TWebDataInput Members 2088
TWebDataInput Methods 2089

TWebDataInput Properties 2093
TWebDisplay Class 2095
TWebDisplay Members 2095
TWebDisplay.Create Constructor 2095
TWebDisplay Methods 2096
TWebDisplay Properties 2097
TWebDisplay Events 2098
TWebDisplayReferenceGroup Class 2098
TWebDisplayReferenceGroup Members 2099
TWebDisplayReferenceGroup.Create Constructor 2100
TWebDisplayReferenceGroup.Destroy Destructor 2100
TWebDisplayReferenceGroup Methods 2100
TWebDisplayReferenceGroup Properties 2101
TWebForm Class 2102
TWebForm Members 2103
TWebForm.Create Constructor 2104
TWebForm.Destroy Destructor 2104
TWebForm Methods 2104
TWebForm Properties 2110
TWebListInput Class 2113
TWebListInput Members 2114
TWebListInput.Create Constructor 2116
TWebListInput.Destroy Destructor 2116
TWebListInput Methods 2116
TWebListInput Properties 2119
TWebRadioGroupInput Class 2122
TWebRadioGroupInput Members 2123
TWebRadioGroupInput.Create Constructor 2125
TWebRadioGroupInput Methods 2125
TWebRadioGroupInput Properties 2127
TWebSelectOptionsInput Class 2128
TWebSelectOptionsInput Members 2129
TWebSelectOptionsInput.Create Constructor 2131
TWebSelectOptionsInput Methods 2131
TWebSelectOptionsInput Properties 2132
TWebStatus Class 2133
TWebStatus Members 2134
TWebStatus.Create Constructor 2135
TWebStatus Methods 2135

TWebStatus Properties 2137
TWebTextAreaInput Class 2138
TWebTextAreaInput Members 2140
TWebTextAreaInput.Create Constructor 2141
TWebTextAreaInput Methods 2141
TWebTextAreaInput Properties 2143
TWebTextInput Class 2145
TWebTextInput Members 2146
TWebTextInput.Create Constructor 2147
TWebTextInput Methods 2148
TWebTextInput Properties 2149
TXMLButton Class 2151
TXMLButton Members 2152
TXMLButton.Create Constructor 2153
TXMLButton.Destroy Destructor 2153
TXMLButton Methods 2153
TXMLButton Properties 2155
TXMLData Class 2157
TXMLData Members 2157
TXMLData.Create Constructor 2158
TXMLData Methods 2158
TXMLData Properties 2160
TXMLData Events 2161
TXMLDataParent Class 2161
TXMLDataParent Members 2162
TXMLDataParent Methods 2162
TXMLDataParent Properties 2164
TXMLDataSet Class 2164
TXMLDataSet Members 2165
TXMLDataSet.Create Constructor 2165
TXMLDataSet.Destroy Destructor 2166
TXMLDataSet Methods 2166
TXMLDataSet Properties 2170
TXMLDataSetParent Class 2172
TXMLDataSetParent Members 2172
TXMLDataSetParent Methods 2173
TXMLDataSetParent Properties 2175
TXMLDisplayGroup Class 2176
TXMLDisplayGroup Members 2177

TXMLDisplayGroup.Create Constructor 2178
TXMLDisplayGroup.Destroy Destructor 2178
TXMLDisplayGroup Methods 2179
TXMLDisplayGroup Properties 2186
TXMLDisplayParent Class 2188
TXMLDisplayParent Members 2189
TXMLDisplayParent.Create Constructor 2189
TXMLDisplayParent Methods 2189
TXMLDisplayParent Properties 2190
TXMLDisplayReferenceButton Class 2191
TXMLDisplayReferenceButton Members 2192
TXMLDisplayReferenceButton.Create Constructor 2193
TXMLDisplayReferenceButton.Destroy Destructor 2193
TXMLDisplayReferenceButton Methods 2193
TXMLDisplayReferenceButton Properties 2195
TXMLDisplayReferenceGroup Class 2197
TXMLDisplayReferenceGroup Members 2197
TXMLDisplayReferenceGroup Methods 2198
TXMLDisplayReferenceGroup Properties 2199

Interfaces 2200

IDataSetComponent Interface 2200
IDataSetComponent Members 2201
IDataSetComponent Methods 2201
IDataSetComponent Properties 2202
IDataSetField Interface 2204
IDataSetField Members 2204
IDataSetField Methods 2204
IDataSetField Properties 2205
IDataSetFields Interface 2206
IDataSetFields Members 2206
IDataSetFields Methods 2206
IDataSetFields Properties 2208
IFormatColumn Interface 2208
IFormatColumn Members 2208
IFormatColumn Methods 2209
IHTMLField Interface 2209
IHTMLField Members 2210
IHTMLField Methods 2210
IHTMLField Properties 2210

IHTMLForm Interface 2211
IHTMLForm Members 2211
IHTMLForm Methods 2212
IHTMLForm Properties 2213
IInetXWebDataSet Interface 2213
IInetXWebDataSet Members 2214
IInetXWebDataSet Methods 2214
IQueryField Interface 2217
IQueryField Members 2217
IQueryField Methods 2218
IQueryField Properties 2219
IQueryFields Interface 2220
IQueryFields Members 2220
IQueryFields Methods 2220
IQueryForm Interface 2221
IRestoreDefaults Interface 2221
IRestoreDefaults Members 2222
IRestoreDefaults Methods 2222
IStatusField Interface 2223
IValidateField Interface 2223
IValidateField Members 2223
IValidateField Methods 2223
IValidateFields Interface 2224
IValidateFields Members 2224
IValidateFields Methods 2225
IValidateFields Properties 2225
IWebDisplay Interface 2226
IWebDisplay Members 2226
IWebDisplay Methods 2226
IWebDisplay Properties 2227
IWebDisplayReference Interface 2227
IWebDisplayReference Members 2228
IWebDisplayReference Methods 2228
IWebDisplayReference Properties 2228
IXMLDisplay Interface 2229
IXMLDisplay Members 2229
IXMLDisplay Methods 2229
IXMLDisplay Properties 2230

Functions 2232

MidItems.AddBoolAttrib Function 2232
MidItems.AddCustomAttrib Function 2232
MidItems.AddIntAttrib Function 2233
MidItems.AddQuotedAttrib Function 2233
MidItems.AddStringAttrib Function 2233
MidItems.CanAddClassHelper Function 2234
MidItems.CreateDefaultButtonClasses Function 2234
MidItems.DeclareSubmitForm Function 2234
MidItems.FindDispatcher Function 2235
MidItems.FindProducer Function 2235
MidItems.FormatColumnData Function 2235
MidItems.FormatColumnHeading Function 2236
MidItems.GetItemValuesFromDataSet Function 2236

Structs, Records, Enums **2237**
MidItems.TCaptionPosition Enumeration 2237
MidItems.TFormMethod Enumeration 2238
MidItems.TLayoutState Enumeration 2238
MidItems.TTextAreaWrap Enumeration 2239

Types **2240**
MidItems.IMidasWebDataSet Type 2240
MidItems.TWebButtonClass Type 2240
MidItems.TWebDataDisplayClass Type 2241
MidItems.TWebDataInputClass Type 2241
MidItems.TWebStatusClass Type 2241

Constants **2242**
MidItems.FormMethodNames Constant 2242
MidItems.HTMLAlignValues Constant 2243
MidItems.HTMLVAlignValues Constant 2243
MidItems.sActionAttr Constant 2243
MidItems.sAlignAttr Constant 2244
MidItems.sBgColorAttr Constant 2244
MidItems.sBorderAttr Constant 2244
MidItems.sCellPaddingAttr Constant 2245
MidItems.sCellSpacingAttr Constant 2245
MidItems.sCheckedAttr Constant 2245
MidItems.sClassAttr Constant 2246
MidItems.sColsAttr Constant 2246
MidItems.sColSpanAttr Constant 2246

MidItems.sEncTypeAttr Constant 2247
MidItems.sImageHeightAttr Constant 2247
MidItems.sImageWidthAttr Constant 2247
MidItems.sMaxLengthAttr Constant 2248
MidItems.sMethodAttr Constant 2248
MidItems.sMultipleAttr Constant 2248
MidItems.sNameAttr Constant 2249
MidItems.sOnClickAttr Constant 2249
MidItems.sReadOnlyAttr Constant 2249
MidItems.sRowsAttr Constant 2250
MidItems.sSelectedAttr Constant 2250
MidItems.sSizeAttr Constant 2250
MidItems.sStyleAttr Constant 2251
MidItems.sTabIndexAttr Constant 2251
MidItems.sTypeAttr Constant 2251
MidItems.sVAlignAttr Constant 2252
MidItems.sValueAttr Constant 2252
MidItems.sWidthAttr Constant 2252
MidItems.sWrapAttr Constant 2253
MidItems.TextAreaWrap Constant 2253

MidProd Namespace 2255

Classes 2257

TBaseInetXPageProducer Class 2257
TBaseInetXPageProducer Members 2258
TBaseInetXPageProducer.Create Constructor 2259
TBaseInetXPageProducer.Destroy Destructor 2259
TBaseInetXPageProducer Methods 2260
TBaseInetXPageProducer Properties 2270
TBaseInetXPageProducer Events 2273
TCustomInetXPageProducer Class 2275
TCustomInetXPageProducer Members 2276
TCustomInetXPageProducer.Create Constructor 2278
TCustomInetXPageProducer.Destroy Destructor 2278
TCustomInetXPageProducer Methods 2279
TCustomInetXPageProducer Properties 2281
TInetXPageElements Class 2283
TInetXPageElements Members 2283

TInetXPageElements Fields 2283
TInetXPageElements Methods 2285
TInetXPageProducer Class 2285
TInetXPageProducer Members 2286
TInetXPageProducer Properties 2287
TInetXPageProducer Events 2292
TMidasPageProducer Class 2295

Functions **2296**
MidProd.AddStdIncludes Function 2296
MidProd.PathInfoToRelativePath Function 2296

Structs, Records, Enums **2297**
MidProd.TStdInclude Enumeration 2297

Types **2298**
MidProd.TBaseMidasPageProducer Type 2298
MidProd.TCustomMidasPageProducer Type 2298
MidProd.TMidasPageElements Type 2299
MidProd.TStdIncludes Type 2299
MidProd.TXMLDataEvent Type 2299

Constants **2301**
MidProd.sXMLBroker Constant 2301
MidProd.sXMLReadyVar Constant 2301

MPlayer Namespace 2303

Classes **2305**
EMCIDeviceError Class 2305
TMediaPlayer Class 2305
TMediaPlayer Members 2309
TMediaPlayer.Create Constructor 2311
TMediaPlayer.Destroy Destructor 2311
TMediaPlayer Methods 2312
TMediaPlayer Properties 2331
TMediaPlayer Events 2366

Structs, Records, Enums **2376**
MPlayer.TMPBtnType Enumeration 2376
MPlayer.TMPButton Record 2377
MPlayer.TMPDevCaps Enumeration 2378

MPlayer.TMPDeviceTypes Enumeration 2378
MPlayer.TMPGlyph Enumeration 2380
MPlayer.TMPModes Enumeration 2380
MPlayer.TMPNotifyValues Enumeration 2381
MPlayer.TMPTimeFormats Enumeration 2382

Types **2384**
MPlayer.EMPNotify Type 2384
MPlayer.EMPPostNotify Type 2384
MPlayer.TButtonSet Type 2385
MPlayer.TMediaDevice Type 2385
MPlayer.TMPDevCapsSet Type 2385

Mtsobj Namespace 2387

Classes **2388**
TMtsAutoObject Class 2388
TMtsAutoObject Members 2388
TMtsAutoObject Methods 2389
TMtsAutoObject Properties 2394

MtsRdm Namespace 2397

Classes **2398**
TMtsDataModule Class 2398
TMtsDataModule Members 2399
TMtsDataModule.Create Constructor 2399
TMtsDataModule Methods 2400
TMtsDataModule Properties 2410
TMtsDataModule Events 2412

Mtx Namespace 2415

Interfaces **2417**
IEnumNames Interface 2417
IEnumNames Members 2417
IEnumNames Methods 2418
IGetContextProperties Interface 2419
IGetContextProperties Members 2419
IGetContextProperties Methods 2419
IObjectContext Interface 2420

IObjectContext Members 2420
IObjectContext Methods 2421
IObjectContextActivity Interface 2423
IObjectContextActivity Members 2423
IObjectContextActivity Methods 2423
IObjectControl Interface 2424
IObjectControl Members 2424
IObjectControl Methods 2424
ISecurityProperty Interface 2425
ISecurityProperty Members 2426
ISecurityProperty Methods 2426
ISharedProperty Interface 2427
ISharedProperty Members 2427
ISharedProperty Methods 2428
ISharedProperty Properties 2428
ISharedPropertyGroup Interface 2429
ISharedPropertyGroup Members 2429
ISharedPropertyGroup Methods 2429
ISharedPropertyGroup Properties 2431
ISharedPropertyGroupManager Interface 2431
ISharedPropertyGroupManager Members 2432
ISharedPropertyGroupManager Methods 2432
ISharedPropertyGroupManager Properties 2433
ITransactionContext Interface 2433
ITransactionContext Members 2434
ITransactionContext Methods 2434
ITransactionContextEx Interface 2435
ITransactionContextEx Members 2435
ITransactionContextEx Methods 2435

Functions **2437**

Mtx.CreateSharedPropertyGroup Function 2437
Mtx.CreateSharedPropertyGroupManager Function 2438
Mtx.CreateTransactionContext Function 2438
Mtx.CreateTransactionContextEx Function 2439
Mtx.GetObjectContext Function 2440
Mtx.SafeRef Function 2440

Constants **2442**

Mtx.CLASS_SharedPropertyGroupManager Constant 2442

Mtx.CLASS_TransactionContext Constant 2442
Mtx.CLASS_TransactionContextEx Constant 2443
Mtx.LockMethod Constant 2443
Mtx.LockSetGet Constant 2443
Mtx.mtsErrCtxAborted Constant 2444
Mtx.mtsErrCtxAborting Constant 2444
Mtx.mtsErrCtxActivityTimeout Constant 2444
Mtx.mtsErrCtxNoContext Constant 2445
Mtx.mtsErrCtxNoSecurity Constant 2445
Mtx.mtsErrCtxNotRegistered Constant 2445
Mtx.mtsErrCtxOldReference Constant 2446
Mtx.mtsErrCtxRoleNotFound Constant 2446
Mtx.mtsErrCtxTMNotAvailable Constant 2446
Mtx.mtsErrCtxWrongThread Constant 2447
Mtx.Process Constant 2447
Mtx.Standard Constant 2447

ObjAuto Namespace 2449

Interfaces 2451
IMethodHandler Interface 2451
IMethodHandler Members 2451
IMethodHandler Methods 2451

Functions 2453
ObjAuto.CreateMethodPointer Function 2453
ObjAuto.CreateMethodPointer Function 2453
ObjAuto.GetInvokeInstance Function 2454
ObjAuto.GetMethodInfo Function 2454
ObjAuto.GetMethods Function 2454
ObjAuto.ObjectInvoke Function 2455
ObjAuto.ReleaseMethodPointer Function 2455

Structs, Records, Enums 2456
ObjAuto.TCallingConvention Enumeration 2456
ObjAuto.TMethodInfoHeader Record 2456
ObjAuto.TParameters Record 2457
ObjAuto.TParamInfo Record 2457
ObjAuto.TReturnInfo Record 2458

Types 2459

ObjAuto.PMethodInfoHeader Type 2459
ObjAuto.PParameters Type 2459
ObjAuto.PParamInfo Type 2460
ObjAuto.PPointer Type 2460
ObjAuto.PReturnInfo Type 2460
ObjAuto.PWord Type 2461
ObjAuto.TDynamicInvokeEvent Type 2461
ObjAuto.TMethodInfoArray Type 2461
ObjAuto.TParamFlags Type 2462

Constants **2463**
ObjAuto.paEAX Constant 2463
ObjAuto.paECX Constant 2463
ObjAuto.paEDX Constant 2464
ObjAuto.paStack Constant 2464

ObjBrkr Namespace 2465

Classes **2466**
EBrokerException Class 2466
TServerCollection Class 2466
TServerCollection Members 2467
TServerCollection.Create Constructor 2467
TServerCollection Methods 2468
TServerCollection Properties 2469
TServerItem Class 2470
TServerItem Members 2471
TServerItem.Create Constructor 2471
TServerItem Methods 2472
TServerItem Properties 2472
TSimpleObjectBroker Class 2475
TSimpleObjectBroker Members 2475
TSimpleObjectBroker.Create Constructor 2476
TSimpleObjectBroker.Destroy Destructor 2476
TSimpleObjectBroker Methods 2476
TSimpleObjectBroker Properties 2480

ObjComAuto Namespace 2483

Classes **2484**

TObjectDispatch Class 2484
TObjectDispatch Members 2484
TObjectDispatch.Create Constructor 2485
TObjectDispatch.Destroy Destructor 2485
TObjectDispatch Methods 2485
TObjectDispatch Properties 2488

Functions 2489
ObjComAuto.CreateMethodPointer Function 2489
ObjComAuto.CreateMethodPointer Function 2489
ObjComAuto.ReleaseMethodPointer Function 2490

Structs, Records, Enums 2491
ObjComAuto.TDispatchInfo Record 2491
ObjComAuto.TDispatchKind Enumeration 2491

Types 2493
ObjComAuto.TDispatchInfos Type 2493

OleAuto Namespace 2495

Classes 2497
EOleError Class 2497
EOleException Class 2497
EOleException Members 2498
EOleException.Create Constructor 2498
EOleException Properties 2498
EOleSysError Class 2499
EOleSysError Members 2499
EOleSysError.Create Constructor 2500
EOleSysError Properties 2500
TAutoDispatch Class 2500
TAutoDispatch Members 2501
TAutoDispatch.Create Constructor 2501
TAutoDispatch Methods 2501
TAutoDispatch Properties 2504
TAutomation Class 2504
TAutomation Members 2504
TAutomation.Create Constructor 2505
TAutomation.Destroy Destructor 2505
TAutomation Methods 2505

TAutomation Properties 2506
TAutomation Events 2507
TAutoObject Class 2507
TAutoObject Members 2508
TAutoObject.Create Constructor 2508
TAutoObject.Destroy Destructor 2508
TAutoObject Methods 2509
TAutoObject Properties 2510
TRegistryClass Class 2511
TRegistryClass Members 2511
TRegistryClass.Create Constructor 2512
TRegistryClass.Destroy Destructor 2512
TRegistryClass Methods 2512

Functions **2513**
OleAuto.ClassIDToProgID Function 2513
OleAuto.ClassIDToString Function 2513
OleAuto.CreateOleObject Function 2514
OleAuto.DispInvoke Function 2515
OleAuto.DispInvokeError Function 2516
OleAuto.DllCanUnloadNow Function 2516
OleAuto.DllGetClassObject Function 2516
OleAuto.DllRegisterServer Function 2517
OleAuto.DllUnregisterServer Function 2517
OleAuto.GetActiveOleObject Function 2517
OleAuto.OleCheck Function 2518
OleAuto.OleError Function 2518
OleAuto.ProgIDToClassID Function 2518
OleAuto.StringToClassID Function 2519
OleAuto.VarFromInterface Function 2519
OleAuto.VarToAutoObject Function 2519
OleAuto.VarToInterface Function 2520

Structs, Records, Enums **2521**
OleAuto.TAutoClassInfo Record 2521
OleAuto.TAutoClassInstancing Enumeration 2521
OleAuto.TStartMode Enumeration 2522

Types **2523**
OleAuto.TAutoClass Type 2523
OleAuto.TLastReleaseEvent Type 2523

Variables **2524**
OleAuto.Automation Variable 2524

Constants **2525**
OleAuto.MaxDispArgs Constant 2525

OleCtnrs Namespace 2527

Classes **2529**
TOleContainer Class 2529
TOleContainer Members 2530
TOleContainer.Create Constructor 2532
TOleContainer.Destroy Destructor 2533
TOleContainer Methods 2533
TOleContainer Properties 2551
TOleContainer Events 2582
TOleForm Class 2613
TOleForm Members 2613
TOleForm.Create Constructor 2614
TOleForm.Destroy Destructor 2614

Interfaces **2615**
IVCLFrameForm Interface 2615
IVCLFrameForm Members 2615
IVCLFrameForm Methods 2615

Functions **2617**
OleCtnrs.DestroyMetaPict Function 2617

Structs, Records, Enums **2618**
OleCtnrs.TAutoActivate Enumeration 2618
OleCtnrs.TCreateInfo Record 2618
OleCtnrs.TCreateType Enumeration 2619
OleCtnrs.TObjectState Enumeration 2620
OleCtnrs.TSizeMode Enumeration 2621
OleCtnrs.TVerbInfo Record 2621

Types **2622**
OleCtnrs.TObjectMoveEvent Type 2622

Constants **2623**
OleCtnrs.ovDiscardUndoState Constant 2623
OleCtnrs.ovHide Constant 2624

OleCtnrs.ovInPlaceActivate Constant 2624
OleCtnrs.ovOpen Constant 2625
OleCtnrs.ovPrimary Constant 2625
OleCtnrs.ovShow Constant 2626
OleCtnrs.ovUIActivate Constant 2627

OleCtrls Namespace 2629

Classes 2631
EOleCtrlError Class 2631
TEnumPropDesc Class 2631
TEnumPropDesc Members 2632
TEnumPropDesc.Create Constructor 2632
TEnumPropDesc.Destroy Destructor 2633
TEnumPropDesc Methods 2633
TEventDispatch Class 2634
TEventDispatch Members 2635
TEventDispatch.Create Constructor 2635
TEventDispatch Methods 2635
TEventDispatch Properties 2638
TOleControl Class 2638
TOleControl Members 2639
TOleControl.Create Constructor 2643
TOleControl.Destroy Destructor 2643
TOleControl Methods 2643
TOleControl Properties 2695
TOleControl Events 2698

Functions 2699
OleCtrls.EmptyParam Function 2699
OleCtrls.FontToOleFont Function 2699
OleCtrls.OleFontToFont Function 2700

Structs, Records, Enums 2701
OleCtrls.TControlData Record 2701
OleCtrls.TControlData2 Record 2702
OleCtrls.TEnumValue Record 2702

Types 2704
OleCtrls.PControlData Type 2704
OleCtrls.PControlData2 Type 2704

OleCtrls.PEnumValueList Type 2705
OleCtrls.TEnumValueList Type 2705
OleCtrls.TOleEnum Type 2705
OleCtrls.TServiceQuery Type 2706

OleServer Namespace 2707

Classes **2708**
TOleServer Class 2708
TOleServer Members 2708
TOleServer.Create Constructor 2709
TOleServer.Destroy Destructor 2710
TOleServer Methods 2711
TOleServer Properties 2717
TServerEventDispatch Class 2720
TServerEventDispatch Members 2720
TServerEventDispatch.Create Constructor 2720
TServerEventDispatch Methods 2721
TServerEventDispatch Properties 2724

Structs, Records, Enums **2725**
OleServer.TConnectKind Enumeration 2725
OleServer.TServerData Record 2725

Types **2727**
OleServer.PServerData Type 2727
OleServer.TVariantArray Type 2727

OPConvert Namespace 2729

Interfaces **2730**
IOPConvert Interface 2730
IOPConvert Members 2731
IOPConvert Methods 2731
IOPConvert Properties 2738

Structs, Records, Enums **2740**
OPConvert.TSOAPConvertOption Enumeration 2740

Types **2743**
OPConvert.TSOAPConvertOptions Type 2743

OPToSOAPDomConv Namespace 2745

Classes **2747**
ESOAPDomConvertError Class 2747
TOPToSoapDomConvert Class 2747
TOPToSoapDomConvert Members 2748
TOPToSoapDomConvert.Create Constructor 2751
TOPToSoapDomConvert.Destroy Destructor 2751
TOPToSoapDomConvert Methods 2751
TOPToSoapDomConvert Properties 2756
TSOAPDomConv Class 2758
TSOAPDomConv Members 2759
TSOAPDomConv.Create Constructor 2760
TSOAPDomConv.Destroy Destructor 2761
TSOAPDomConv Methods 2761
TSOAPDomConv Properties 2781
TSOAPDomConv Events 2782

Functions **2783**
OPToSOAPDomConv.GetOrdPropEx Function 2783

Structs, Records, Enums **2784**
OPToSOAPDomConv.ConvNodeState Enumeration 2784
OPToSOAPDomConv.TMultiRefNodeElem Record 2784
OPToSOAPDomConv.TMultiRefNodeMapElem Record 2785
OPToSOAPDomConv.TSOAPArrayElemDesc Record 2785

Types **2786**
OPToSOAPDomConv.TMemberDataNotReceivedEvent Type 2786
OPToSOAPDomConv.TMultiRefNodeMap Type 2787
OPToSOAPDomConv.TMultiRefNodes Type 2787
OPToSOAPDomConv.TSOAPArrayDesc Type 2787
OPToSOAPDomConv.TUnhandledNodeEvent Type 2788
OPToSOAPDomConv.TXMLNodeArray Type 2788

Variables **2789**
OPToSOAPDomConv.DefArrayElemName Variable 2789

Constants **2790**
OPToSOAPDomConv.SVarArrayType Constant 2790

OrbPas Namespace 2791

Interfaces **2793**
IBOA Interface 2793
IBOA Members 2793
IBOA Methods 2794
ICorbaObj Interface 2795
ICorbaObj Members 2795
ICorbaObj Methods 2795
IMarshalInBuffer Interface 2796
IMarshalInBuffer Members 2796
IMarshalInBuffer Methods 2797
IMarshalOutBuffer Interface 2800
IMarshalOutBuffer Members 2801
IMarshalOutBuffer Methods 2801
IORB Interface 2805
IORB Members 2805
IORB Methods 2805
ISkeleton Interface 2809
ISkeleton Members 2809
ISkeleton Methods 2810
ISkeletonObject Interface 2810
ISkeletonObject Members 2811
ISkeletonObject Methods 2811
IStub Interface 2812
IStub Members 2812
IStub Methods 2812
IStubObject Interface 2815
IStubObject Members 2815
IStubObject Methods 2815
ITypeCode Interface 2816
ITypeCode Members 2816
ITypeCode Methods 2817

Functions **2821**
OrbPas.AnyToVariant Function 2821
OrbPas.BindStub Function 2821
OrbPas.CorbaAnyType Function 2822
OrbPas.CorbaDuplicateAny Function 2822

OrbPas.CorbaReleaseAny Function 2822
OrbPas.CorbaStringFree Function 2823
OrbPas.CorbaWStringFree Function 2823
OrbPas.CreateSkeleton Function 2823
OrbPas.CreateUserException Function 2824
OrbPas.InitORB Function 2824
OrbPas.RegisterUserException Function 2824
OrbPas.SequenceToVariant Function 2825
OrbPas.UnRegisterUserException Function 2825
OrbPas.VariantToAny Function 2825

Structs, Records, Enums **2826**
OrbPas.TCKind Enumeration 2826
OrbPas.TStructMember Record 2827

Types **2828**
OrbPas.CorbaBoolean Type 2828
OrbPas.CorbaULong Type 2828
OrbPas.PCorbaAny Type 2829
OrbPas.PCorbaInterfaceDef Type 2829
OrbPas.PCorbaObject Type 2829
OrbPas.PCorbaOrb Type 2830
OrbPas.PExceptionDescription Type 2830
OrbPas.PMarshalInbuffer Type 2830
OrbPas.PMarshalOutbuffer Type 2831
OrbPas.PUserExceptionProxy Type 2831
OrbPas.TArgv Type 2831
OrbPas.TCopyUserExceptionProc Type 2832
OrbPas.TStructMembers Type 2832
OrbPas.TThrowUserExceptionProc Type 2832
OrbPas.TUserExceptionFactoryProc Type 2833

Outline Namespace 2835

Classes **2837**
EOutlineError Class 2837
OutlineError Class 2838
TCustomOutline Class 2838
TCustomOutline Members 2839
TCustomOutline.Create Constructor 2840
TCustomOutline.Destroy Destructor 2841

TCustomOutline Methods 2841
TCustomOutline Properties 2862
TCustomOutline Events 2872
TOutline Class 2874
TOutline Members 2874
TOutline Properties 2877
TOutline Events 2905
TOutlineNode Class 2940
TOutlineNode Members 2941
TOutlineNode.Create Constructor 2942
TOutlineNode.Destroy Destructor 2943
TOutlineNode Methods 2943
TOutlineNode Properties 2951

Structs, Records, Enums 2957
Outline.TAttachMode Enumeration 2957
Outline.TOutlineBitmap Enumeration 2958
Outline.TOutlineNodeCompare Enumeration 2958
Outline.TOutlineOption Enumeration 2959
Outline.TOutlineStyle Enumeration 2959
Outline.TOutlineType Enumeration 2960

Types 2962
Outline.EOutlineChange Type 2962
Outline.TBitmapArray Type 2962
Outline.TBitmapArrayRange Type 2963
Outline.TChangeRange Type 2963
Outline.TOutlineBitmaps Type 2963
Outline.TOutlineOptions Type 2964

PagItems Namespace 2965

Classes 2966
TBasePageItemsProducer Class 2966
TBasePageItemsProducer Members 2967
TBasePageItemsProducer.Create Constructor 2967
TBasePageItemsProducer.Destroy Destructor 2967
TBasePageItemsProducer Methods 2968
TBasePageItemsProducer Properties 2969
TPageItemsProducer Class 2970
TPageItemsProducer Members 2971

TPageItemsProducer.Create Constructor 2971
TPageItemsProducer.Destroy Destructor 2971
TPageItemsProducer Methods 2972
TPageItemsProducer Properties 2972

PlatformDefaultStyleActnCtrls Namespace 2975

Classes **2976**
TPlatformDefaultStyleActionBars Class 2976
TPlatformDefaultStyleActionBars Members 2976
TPlatformDefaultStyleActionBars Methods 2976

Variables **2979**
PlatformDefaultStyleActnCtrls.PlatformDefaultStyle Variable 2979

pngimage Namespace 2981

Classes **2984**
EPNGCannotChangeTransparent Class 2984
EPNGCouldNotLoadResource Class 2985
EPngError Class 2985
EPNGHeaderNotPresent Class 2985
EPNGIHDRNotFirst Class 2986
EPngInvalidCRC Class 2986
EPNGInvalidFileHeader Class 2987
EPngInvalidIHDR Class 2987
EPNGInvalidNewSize Class 2987
EPNGInvalidPalette Class 2988
EPNGInvalidSpec Class 2988
EPNGMissingMultipleIDAT Class 2988
EPNGMissingPalette Class 2989
EPNGNoImageData Class 2989
EPNGNotExists Class 2990
EPNGOutMemory Class 2990
EPNGSizeExceeds Class 2990
EPngUnexpectedEnd Class 2991
EPNGUnknownCompression Class 2991
EPNGUnknownCriticalChunk Class 2991
EPNGUnknownInterlace Class 2992
EPNGZLIBError Class 2992

Exception Class 2993
Exception Members 2993
Exception.Create Constructor 2993
TFileStream Class 2993
TFileStream Members 2994
TFileStream.Create Constructor 2994
TFileStream.Destroy Destructor 2994
TFileStream Methods 2995
TPngImage Class 2996
TPNGList Class 2997
TPNGList Members 2997
TPNGList Methods 2998
TPNGList Properties 2999
TPNGPointerList Class 2999
TPNGPointerList Members 3000
TPNGPointerList.Create Constructor 3000
TPNGPointerList.Destroy Destructor 3000
TPNGPointerList Methods 3001
TPNGPointerList Properties 3002
TResourceStream Class 3003
TResourceStream Members 3003
TResourceStream.Create Constructor 3003
TResourceStream Methods 3004
TStream Class 3005
TStream Members 3005
TStream Methods 3006
TStream Properties 3008

Functions 3009

pngimage.AddtEXt Function 3009
pngimage.AddzTXt Function 3010
pngimage.Assign Function 3010
pngimage.AssignHandle Function 3010
pngimage.AssignPNG Function 3011
pngimage.CreateAlpha Function 3011
pngimage.DoSetPalette Function 3011
pngimage.Draw Function 3012
pngimage.DrawPartialTrans Function 3012
pngimage.DrawUsingPixelInformation Function 3012
pngimage.GetAlphaScanline Function 3013

pngimage.GetEmpty Function 3013
pngimage.GetLibraryVersion Function 3013
pngimage.GetPalette Function 3014
pngimage.GetPixelInformation Function 3014
pngimage.GetPixels Function 3014
pngimage.GetScanline Function 3015
pngimage.GetSupportsPartialTransparency Function 3015
pngimage.GetTransparencyMode Function 3015
pngimage.GetTransparentColor Function 3016
pngimage.GetWidth Function 3016
pngimage.HasPixelInformation Function 3016
pngimage.RaiseError Function 3017
pngimage.RemoveTransparency Function 3017
pngimage.Resize Function 3017
pngimage.SetMaxIdatSize Function 3018
pngimage.SetPixels Function 3018
pngimage.SetTransparentColor Function 3018
pngimage.SetWidth Function 3019

Structs, Records, Enums 3020

pngimage.TFileStreamMode Enumeration 3020
pngimage.TFilter Enumeration 3020
pngimage.TInterlaceMethod Enumeration 3021
pngimage.TMAXBITMAPINFO Record 3021
pngimage.TPNGTransparencyMode Enumeration 3022
pngimage.TRGBPixel Record 3022

Types 3023

pngimage.ExceptClass Type 3023
pngimage.pByteArray Type 3023
pngimage.pCardinal Type 3024
pngimage.pPointerArray Type 3024
pngimage.pRGBLine Type 3024
pngimage.pRGBPixel Type 3025
pngimage.TBitmap Type 3025
pngimage.TByteArray Type 3025
pngimage.TCanvas Type 3026
pngimage.TChunkClass Type 3026
pngimage.TColor Type 3026
pngimage.TCompressionLevel Type 3027

pngimage.TFileStreamModeSet Type 3027
pngimage.TFilters Type 3027
pngimage.TPersistent Type 3028
pngimage.TPointerArray Type 3028
pngimage.TRGBLine Type 3028

Constants **3029**
pngimage.COLOR_GRAYSCALE Constant 3029
pngimage.COLOR_GRAYSCALEALPHA Constant 3029
pngimage.COLOR_PALETTE Constant 3030
pngimage.COLOR_RGB Constant 3030
pngimage.COLOR_RGBALPHA Constant 3030
pngimage.FILTER_AVERAGE Constant 3031
pngimage.FILTER_NONE Constant 3031
pngimage.FILTER_PAETH Constant 3031
pngimage.FILTER_SUB Constant 3032
pngimage.FILTER_UP Constant 3032
pngimage.Index Constant 3032
pngimage.LibraryVersion Constant 3033
pngimage.soFromBeginning Constant 3033
pngimage.soFromCurrent Constant 3033
pngimage.soFromEnd Constant 3034
pngimage.Z_FINISH Constant 3034
pngimage.Z_NO_FLUSH Constant 3034
pngimage.Z_STREAM_END Constant 3035
pngimage.ZLIBErrors Constant 3035

Printers Namespace 3037

Classes **3038**
EPrinter Class 3038
TPrinter Class 3038
TPrinter Members 3039
TPrinter.Create Constructor 3040
TPrinter.Destroy Destructor 3040
TPrinter Methods 3040
TPrinter Properties 3050

Functions **3066**
Printers.AssignPrn Function 3066
Printers.Printer Function 3066

Printers.SetPrinter Function 3068

Structs, Records, Enums **3070**

Printers.TPrinterCapability Enumeration 3070

Printers.TPrinterOrientation Enumeration 3070

Printers.TPrinterState Enumeration 3071

Types **3072**

Printers.TPrinterCapabilities Type 3072

Provider Namespace 3073

Classes **3075**

EDSWriter Class 3075

EDSWriter Members 3076

EDSWriter.Create Constructor 3076

EDSWriter Properties 3076

TBaseProvider Class 3077

TBaseProvider Members 3078

TBaseProvider.Create Constructor 3079

TBaseProvider.Destroy Destructor 3080

TBaseProvider Methods 3080

TBaseProvider Properties 3087

TBaseProvider Events 3089

TCustomPacketWriter Class 3093

TCustomPacketWriter Members 3093

TCustomPacketWriter.Create Constructor 3094

TCustomPacketWriter.Destroy Destructor 3094

TCustomPacketWriter Fields 3095

TCustomPacketWriter Methods 3095

TCustomPacketWriter Properties 3096

TCustomProvider Class 3096

TCustomProvider Members 3097

TCustomProvider.Create Constructor 3099

TCustomProvider.Destroy Destructor 3099

TCustomProvider Methods 3099

TCustomProvider Properties 3114

TCustomProvider Events 3116

TCustomResolver Class 3125

TCustomResolver Members 3126

TCustomResolver.Create Constructor 3126

TCustomResolver.Destroy Destructor 3127
TCustomResolver Methods 3127
TCustomResolver Properties 3135
TDataPacketWriter Class 3136
TDataPacketWriter Members 3136
TDataPacketWriter.Destroy Destructor 3137
TDataPacketWriter Methods 3138
TDataPacketWriter Properties 3143
TDataPacketWriter Events 3144
TDataSetProvider Class 3145
TDataSetProvider Members 3146
TDataSetProvider.Create Constructor 3148
TDataSetProvider.Destroy Destructor 3148
TDataSetProvider Methods 3148
TDataSetProvider Properties 3159
TDataSetProvider Events 3163
TDataSetResolver Class 3176
TDataSetResolver Members 3177
TDataSetResolver.Create Constructor 3178
TDataSetResolver Methods 3178
TDataSetResolver Properties 3181
TLocalAppServer Class 3182
TLocalAppServer Members 3182
Create Constructor 3183
TLocalAppServer.Destroy Destructor 3184
TLocalAppServer Methods 3184
TPacketDataSet Class 3188
TPacketDataSet Members 3189
TPacketDataSet.Create Constructor 3189
TPacketDataSet Methods 3190
TPacketDataSet Properties 3194
TProvider Class 3196
TSQLResolver Class 3196
TSQLResolver Members 3197
TSQLResolver.Create Constructor 3198
TSQLResolver.Destroy Destructor 3198
TSQLResolver Methods 3199
TSQLResolver Properties 3204
TUpdateTree Class 3205

TUpdateTree Members 3206
TUpdateTree.Create Constructor 3206
TUpdateTree.Destroy Destructor 3207
TUpdateTree Methods 3207
TUpdateTree Properties 3209

Interfaces 3213
IProviderContainer Interface 3213
IProviderContainer Members 3213
IProviderContainer Methods 3213

Functions 3215
Provider.GetObjectProperty Function 3215
Provider.GetStringProperty Function 3215
Provider.VarArrayFromStrings Function 3215
Provider.VarArrayFromWideStrings Function 3216

Structs, Records, Enums 3217
Provider.TGetRecordOption Enumeration 3217
Provider.TProviderOption Enumeration 3218
Provider.TPutFieldInfo Record 3220
Provider.TResolverResponse Enumeration 3221

Types 3222
Provider.PPutFieldInfo Type 3222
Provider.TAfterUpdateRecordEvent Type 3223
Provider.TBeforeCommitEvent Type 3223
Provider.TBeforeUpdateRecordEvent Type 3224
Provider.TDataRequestEvent Type 3224
Provider.TGetDSProps Type 3225
Provider.TGetParamsEvent Type 3225
Provider.TGetRecordOptions Type 3226
Provider.TGetTableNameEvent Type 3226
Provider.TInfoArray Type 3227
Provider.TOnValidate Type 3227
Provider.TProviderDataEvent Type 3228
Provider.TProviderOptions Type 3228
Provider.TPutFieldProc Type 3229
Provider.TResolverErrorEvent Type 3230

Variables 3231
Provider.InformixLob Variable 3231

Constants **3232**
Provider.MetaDataOption Constant 3232
Provider.ResetOption Constant 3233
Provider.XMLOption Constant 3233
Provider.XMLUTF8Option Constant 3234

Registry Namespace 3235

Classes **3236**
ERegistryException Class 3236
TRegIniFile Class 3237
TRegIniFile Members 3237
Create Constructor 3239
TRegIniFile Methods 3240
TRegIniFile Properties 3246
TRegistry Class 3246
TRegistry Members 3249
Create Constructor 3251
TRegistry.Destroy Destructor 3253
TRegistry Methods 3254
TRegistry Properties 3292
TRegistryIniFile Class 3299
TRegistryIniFile Members 3303
Create Constructor 3304
TRegistryIniFile.Destroy Destructor 3304
TRegistryIniFile Methods 3305
TRegistryIniFile Properties 3326

Structs, Records, Enums **3327**
Registry.TRegDataInfo Record 3327
Registry.TRegDataType Enumeration 3327
Registry.TRegKeyInfo Record 3328

ReqFiles Namespace 3331

Classes **3332**
TWebRequestFile Class 3332
TWebRequestFile Members 3333
TWebRequestFile.Create Constructor 3333
TWebRequestFile.Destroy Destructor 3334

TWebRequestFile Methods 3334
TWebRequestFiles Class 3336
TWebRequestFiles Members 3336
TWebRequestFiles.Create Constructor 3336
TWebRequestFiles.Destroy Destructor 3337
TWebRequestFiles Methods 3337
TWebRequestFileStream Class 3339
TWebRequestFileStream Members 3339
TWebRequestFileStream.Create Constructor 3340
TWebRequestFileStream Methods 3340

ReqMulti Namespace 3341

Classes 3342
TMultipartContentParser Class 3342
TMultipartContentParser Members 3342
TMultipartContentParser.Destroy Destructor 3343
TMultipartContentParser Methods 3343

Ribbon Namespace 3347

Classes 3349
ERibbonException Class 3349
TApplicationMenu Class 3350
TApplicationMenu Members 3350
TApplicationMenu.Create Constructor 3351
TApplicationMenu.Destroy Destructor 3351
TApplicationMenu Methods 3351
TApplicationMenu Properties 3353
TCustomActionControlBar Class 3355
TCustomActionControlBar Members 3356
TCustomActionControlBar.Create Constructor 3356
TCustomActionControlBar Methods 3356
TCustomActionControlBar Properties 3357
TCustomRibbon Class 3358
TCustomRibbon Members 3359
TCustomRibbon.Create Constructor 3362
TCustomRibbon.Destroy Destructor 3363
TCustomRibbon Constants 3363
TCustomRibbon Methods 3368

TCustomRibbon Properties 3393
TCustomRibbon Nested Types 3399
TCustomRibbon Events 3400
TCustomRibbonColorMap Class 3401
TCustomRibbonColorMap Members 3402
TCustomRibbonColorMap Properties 3402
TCustomRibbonGroup Class 3405
TCustomRibbonGroup Members 3406
TCustomRibbonGroup.Create Constructor 3407
TCustomRibbonGroup.Destroy Destructor 3408
TCustomRibbonGroup Enumerations 3408
TCustomRibbonGroup Methods 3408
TCustomRibbonGroup Properties 3420
TCustomRibbonGroup Nested Types 3424
TCustomRibbonPage Class 3425
TCustomRibbonPage Members 3425
TCustomRibbonPage.Create Constructor 3426
TCustomRibbonPage.Destroy Destructor 3427
TCustomRibbonPage Constants 3427
TCustomRibbonPage Methods 3428
TCustomRibbonPage Properties 3436
TCustomRibbonQuickAccessPopupToolbar Class 3438
TCustomRibbonQuickAccessPopupToolbar Members 3439
TCustomRibbonQuickAccessPopupToolbar.Create Constructor 3439
TCustomRibbonQuickAccessPopupToolbar Methods 3439
TCustomRibbonQuickAccessToolbar Class 3441
TCustomRibbonQuickAccessToolbar Members 3442
TCustomRibbonQuickAccessToolbar.Create Constructor 3443
TCustomRibbonQuickAccessToolbar Methods 3443
TCustomRibbonQuickAccessToolbar Properties 3451
TCustomRibbonQuickAccessToolbar Nested Types 3453
TCustomTabData Class 3453
TCustomTabData Members 3454
TCustomTabData.Create Constructor 3454
TCustomTabData.Destroy Destructor 3455
TCustomTabData Methods 3455
TCustomTabData Properties 3456
THelpButton Class 3459
THelpButton Members 3460

THelpButton Methods 3460
TOffice2007Button Class 3461
TOffice2007Button Members 3461
TOffice2007Button.Create Constructor 3462
TOffice2007Button.Destroy Destructor 3462
TOffice2007Button Methods 3463
TOffice2007Button Properties 3466
TOffice2007Button Events 3467
TQuickAccessToolbar Class 3469
TQuickAccessToolbar Members 3470
TQuickAccessToolbar.Create Constructor 3470
TQuickAccessToolbar Enumerations 3471
TQuickAccessToolbar Methods 3471
TQuickAccessToolbar Properties 3473
TQuickAccessToolbar Events 3475
TQuickAccessToolbarGroupItem Class 3475
TQuickAccessToolbarGroupItem Members 3475
TQuickAccessToolbarGroupItem.Create Constructor 3476
TQuickAccessToolbarGroupItem.Destroy Destructor 3476
TQuickAccessToolbarGroupItem Properties 3476
TQuickAccessToolbarGroups Class 3477
TQuickAccessToolbarGroups Members 3477
TQuickAccessToolbarGroups.Create Constructor 3478
TQuickAccessToolbarGroups.Destroy Destructor 3478
TQuickAccessToolbarGroups Methods 3478
TRibbon Class 3480
TRibbon Members 3480
TRibbon Properties 3483
TRibbon Events 3498
TRibbonCustomizeDlg Class 3499
TRibbonCustomizeDlg Members 3499
TRibbonCustomizeDlg Methods 3499
TRibbonGroup Class 3500
TRibbonGroup Members 3501
TRibbonGroup Properties 3503
TRibbonGroupCollapsedControl Class 3506
TRibbonGroupCollapsedControl Members 3506
TRibbonGroupCollapsedControl.Create Constructor 3507
TRibbonGroupCollapsedControl.Destroy Destructor 3508

TRibbonGroupCollapsedControl Methods 3508
TRibbonGroupCollapsedControl Properties 3514
TRibbonGroupDockForm Class 3519
TRibbonGroupDockForm Members 3519
TRibbonGroupDockForm.Create Constructor 3519
TRibbonGroupDockForm Methods 3520
TRibbonGroupDockForm Properties 3520
TRibbonGroupScrollButton Class 3521
TRibbonGroupScrollButton Members 3521
TRibbonGroupScrollButton.Create Constructor 3522
TRibbonGroupScrollButton Methods 3522
TRibbonGroupScrollButton Properties 3524
TRibbonLunaColorMap Class 3525
TRibbonLunaColorMap Members 3525
TRibbonLunaColorMap Methods 3526
TRibbonLunaColorMap Properties 3526
TRibbonObsidianColorMap Class 3530
TRibbonObsidianColorMap Members 3531
TRibbonObsidianColorMap Methods 3531
TRibbonObsidianColorMap Properties 3532
TRibbonPage Class 3536
TRibbonPage Members 3537
TRibbonPage Properties 3538
TRibbonQuickAccessPopupToolbar Class 3545
TRibbonQuickAccessToolbar Class 3546
TRibbonQuickAccessToolbar Members 3546
TRibbonQuickAccessToolbar Properties 3548
TRibbonSilverColorMap Class 3549
TRibbonSilverColorMap Members 3549
TRibbonSilverColorMap Methods 3550
TRibbonSilverColorMap Properties 3550
TRibbonTabItem Class 3555
TRibbonTabItem Members 3555
TRibbonTabItem.Destroy Destructor 3555
TRibbonTabItem Methods 3556
TRibbonTabItem Properties 3557
TRibbonTabs Class 3560
TRibbonTabs Members 3560
TRibbonTabs.Create Constructor 3561

TRibbonTabs Methods 3561
TRibbonTabs Properties 3564

Interfaces 3566
IRibbon Interface 3566
IRibbon Members 3566
IRibbon Methods 3567
IRibbon Properties 3567
IRibbonGroup Interface 3568
IRibbonGroup Members 3568
IRibbonGroup Methods 3568
IRibbonQuickAccessToolbar Interface 3569
IRibbonQuickAccessToolbar Members 3570
IRibbonQuickAccessToolbar Methods 3570

Structs, Records, Enums 3571
Ribbon.TCommandType Enumeration 3571
Ribbon.TIconSize Enumeration 3572
Ribbon.TRibbonGroupVariant Enumeration 3572
Ribbon.TRibbonMetric Enumeration 3573
Ribbon.TRibbonScrollButton Enumeration 3573
Ribbon.TTabActiveScrollButton Enumeration 3574
Ribbon.TTabSeparatorState Enumeration 3574

Types 3576
Ribbon.TCustomRibbonClass Type 3576
Ribbon.TRecentItemClickEvent Type 3576
Ribbon.TRibbonScrollButtons Type 3577

RibbonActnCtrls Namespace 3579

Classes 3582
TCustomRibbonComboBox Class 3583
TCustomRibbonComboBox Members 3583
TCustomRibbonComboBox.Create Constructor 3584
TCustomRibbonComboBox.Destroy Destructor 3584
TCustomRibbonComboBox Methods 3584
TCustomRibbonComboBox Properties 3590
TKeyTipList Class 3592
TKeyTipList Members 3592
TKeyTipList.Create Constructor 3593

TKeyTipList.Destroy Destructor 3593
TKeyTipList Methods 3593
TKeyTipList Properties 3594
TKeyTipListEnumerator Class 3596
TKeyTipListEnumerator Members 3596
TKeyTipListEnumerator.Create Constructor 3597
TKeyTipListEnumerator Methods 3597
TKeyTipListEnumerator Properties 3597
TKeyTipMatchItem Class 3598
TKeyTipMatchItem Members 3598
TKeyTipMatchItem.Create Constructor 3599
TKeyTipMatchItem Properties 3599
TKeyTipValidator Class 3600
TKeyTipValidator Members 3601
TKeyTipValidator Methods 3601
TRibbonAddRemoveItem Class 3601
TRibbonAddRemoveItem Members 3602
TRibbonAddRemoveItem Methods 3602
TRibbonApplicationButton Class 3603
TRibbonApplicationButton Members 3603
TRibbonApplicationButton Methods 3603
TRibbonApplicationMenuButton Class 3604
TRibbonApplicationMenuButton Members 3604
TRibbonApplicationMenuButton.Create Constructor 3605
TRibbonApplicationMenuButton.Destroy Destructor 3605
TRibbonApplicationMenuButton Methods 3606
TRibbonApplicationMenuButton Properties 3611
TRibbonApplicationMenuItem Class 3612
TRibbonApplicationMenuItem Members 3613
TRibbonApplicationMenuItem.Create Constructor 3613
TRibbonApplicationMenuItem Methods 3614
TRibbonBaseButtonControl Class 3616
TRibbonBaseButtonControl Members 3617
TRibbonBaseButtonControl.Create Constructor 3617
TRibbonBaseButtonControl.Destroy Destructor 3618
TRibbonBaseButtonControl Methods 3619
TRibbonBaseButtonControl Properties 3625
TRibbonCheckBoxControl Class 3626
TRibbonCheckBoxControl Members 3626

TRibbonCheckBoxControl Methods 3627
TRibbonComboBox Class 3628
TRibbonComboBox Members 3629
TRibbonComboBox Properties 3630
TRibbonComboBox Events 3648
TRibbonComboBoxItem Class 3682
TRibbonComboBoxItem Members 3682
TRibbonComboBoxItem Methods 3682
TRibbonComboControl Class 3684
TRibbonComboControl Members 3684
TRibbonComboControl.Create Constructor 3685
TRibbonComboControl.Destroy Destructor 3686
TRibbonComboControl Methods 3686
TRibbonComboControl Properties 3697
TRibbonCustomControl Class 3698
TRibbonCustomControl Members 3698
TRibbonCustomControl.Create Constructor 3699
TRibbonCustomControl.Destroy Destructor 3699
TRibbonCustomControl Methods 3700
TRibbonCustomControl Properties 3710
TRibbonCustomizeItem Class 3710
TRibbonCustomizeItem Members 3711
TRibbonCustomizeItem Methods 3711
TRibbonCustomStatusControl Class 3713
TRibbonCustomStatusControl Members 3713
TRibbonCustomStatusControl Methods 3714
TRibbonDropDownButton Class 3715
TRibbonDropDownButton Members 3715
TRibbonDropDownButton Methods 3716
TRibbonGallery Class 3720
TRibbonGalleryControl Class 3720
TRibbonGalleryControl Members 3720
TRibbonGalleryControl Methods 3721
TRibbonGalleryControl Properties 3722
TRibbonGalleryControlMenuItem Class 3722
TRibbonGalleryControlMenuItem Members 3723
TRibbonGalleryControlMenuItem Methods 3723
TRibbonGalleryControlMenuItem Properties 3728
TRibbonGalleryItem Class 3728

TRibbonGalleryItem Members 3729
TRibbonGalleryItem Methods 3730
TRibbonGalleryMenuItem Class 3730
TRibbonGalleryMenuItem Members 3731
TRibbonGalleryMenuItem Methods 3731
TRibbonGallerySplitButton Class 3733
TRibbonGallerySplitButton Members 3733
TRibbonGallerySplitButton Methods 3735
TRibbonGripItem Class 3735
TRibbonGripItem Members 3736
TRibbonGripItem.Create Constructor 3736
TRibbonGripItem Methods 3737
TRibbonGripItem Properties 3738
TRibbonGroupDropDown Class 3738
TRibbonGroupDropDown Members 3739
TRibbonGroupDropDown Methods 3740
TRibbonGroupDropDown Properties 3742
TRibbonLargeButtonControl Class 3742
TRibbonLargeButtonControl Members 3742
TRibbonLargeButtonControl Methods 3743
TRibbonLargeSplitButton Class 3745
TRibbonLargeSplitButton Members 3746
TRibbonLargeSplitButton Methods 3747
TRibbonLargeSplitButton Properties 3752
TRibbonMainMenuItem Class 3752
TRibbonMainMenuItem Members 3752
TRibbonMainMenuItem Methods 3753
TRibbonMainMenuSeparator Class 3754
TRibbonMainMenuSeparator Members 3754
TRibbonMainMenuSeparator Methods 3755
TRibbonMenuItem Class 3755
TRibbonMenuItem Members 3755
TRibbonMenuItem.Create Constructor 3756
TRibbonMenuItem Methods 3756
TRibbonMenuItem Properties 3760
TRibbonMenuSeparatorItem Class 3760
TRibbonMenuSeparatorItem Members 3761
TRibbonMenuSeparatorItem Methods 3761
TRibbonMenuSmallButtonControl Class 3762

TRibbonMenuSmallButtonControl Members 3762
TRibbonMenuSmallButtonControl.Create Constructor 3763
TRibbonMenuSmallButtonControl Methods 3764
TRibbonOptionButtonControl Class 3765
TRibbonOptionButtonControl Members 3765
TRibbonOptionButtonControl Methods 3766
TRibbonQATScrollButton Class 3767
TRibbonQATScrollButton Members 3767
TRibbonQATScrollButton Methods 3767
TRibbonQATScrollButton Properties 3770
TRibbonRadioButtonControl Class 3771
TRibbonRadioButtonControl Members 3772
TRibbonRadioButtonControl Methods 3773
TRibbonRecentButtonControl Class 3774
TRibbonRecentButtonControl Members 3774
TRibbonRecentButtonControl Methods 3775
TRibbonRichContentItem Class 3776
TRibbonRichContentItem Members 3776
TRibbonRichContentItem.Create Constructor 3777
TRibbonRichContentItem Methods 3777
TRibbonRichContentMainMenuItem Class 3780
TRibbonRichContentMainMenuItem Members 3780
TRibbonRichContentMainMenuItem.Create Constructor 3781
TRibbonRichContentMainMenuItem Methods 3782
TRibbonSeparator Class 3784
TRibbonSeparator Members 3784
TRibbonSeparator.Create Constructor 3784
TRibbonSeparator Methods 3785
TRibbonSmallButtonControl Class 3787
TRibbonSmallButtonControl Members 3787
TRibbonSmallButtonControl Methods 3788
TRibbonSmallSplitButton Class 3789
TRibbonSmallSplitButton Members 3789
TRibbonSmallSplitButton.Create Constructor 3790
TRibbonSmallSplitButton Fields 3791
TRibbonSmallSplitButton Methods 3792
TRibbonSmallSplitButton Properties 3795
TRibbonSpinButton Class 3796
TRibbonSpinButton Members 3796

TRibbonSpinButton.Create Constructor 3797
TRibbonSpinButton Methods 3798
TRibbonSpinButton Properties 3800
TRibbonSpinButton Events 3820
TRibbonSpinEdit Class 3836
TRibbonSpinEdit Members 3836
TRibbonSpinEdit.Create Constructor 3838
TRibbonSpinEdit.Destroy Destructor 3838
TRibbonSpinEdit Methods 3839
TRibbonSpinEdit Properties 3844
TRibbonSpinEdit Events 3866
TRibbonSplitMenuItem Class 3896
TRibbonSplitMenuItem Members 3897
TRibbonSplitMenuItem.Create Constructor 3898
TRibbonSplitMenuItem Fields 3898
TRibbonSplitMenuItem Methods 3899
TRibbonSplitMenuItem Properties 3901
TRibbonStyleMenuButton Class 3902
TRibbonStyleMenuButton Members 3902
TRibbonStyleMenuButton Methods 3902
TRibbonTextControl Class 3903
TRibbonTextControl Members 3903
TRibbonTextControl Methods 3904
TRibbonToolScrollButton Class 3905
TRibbonToolScrollButton Members 3905
TRibbonToolScrollButton Methods 3906
TTimerSpeedButton Class 3906
TTimerSpeedButton Members 3907
TTimerSpeedButton.Create Constructor 3907
TTimerSpeedButton.Destroy Destructor 3908
TTimerSpeedButton Methods 3908
TTimerSpeedButton Properties 3910

Interfaces 3912
IRibbonBarKeyTip Interface 3912
IRibbonBarKeyTip Members 3912
IRibbonBarKeyTip Methods 3913
IRibbonContainedControl Interface 3914
IRibbonContainedControl Members 3914
IRibbonContainedControl Methods 3915

IRibbonContainedControl Properties 3915
IRibbonControl Interface 3916
IRibbonControl Members 3916
IRibbonControl Methods 3916
IRibbonGroupDropdown Interface 3917
IRibbonGroupDropdown Members 3917
IRibbonGroupDropdown Methods 3917
IRibbonKeyTip Interface 3918
IRibbonKeyTip Members 3919
IRibbonKeyTip Methods 3919
IRibbonKeyTip Properties 3922
IRibbonSplitButton Interface 3922
IRibbonSplitButton Members 3923
IRibbonSplitButton Methods 3923

Functions 3924
RibbonActnCtrls.DrawLine Function 3924
RibbonActnCtrls.IsLeftMouseButtonPressed Function 3924
RibbonActnCtrls.ShowKeyTip Function 3924

Structs, Records, Enums 3926
RibbonActnCtrls.TKeyTipHandled Enumeration 3926
RibbonActnCtrls.TKeyTipMatchType Enumeration 3927
RibbonActnCtrls.TKeyTipPosition Enumeration 3927
RibbonActnCtrls.TKeyTipState Enumeration 3928
RibbonActnCtrls.TQATScrollMode Enumeration 3929
RibbonActnCtrls.TSpinButtonStyle Enumeration 3930

Types 3931
RibbonActnCtrls.TNumGlyphs Type 3931
RibbonActnCtrls.TTimeBtnState Type 3931

Constants 3932
RibbonActnCtrls.UM_FADECONTROL Constant 3932

RibbonActnMenus Namespace 3933

Classes 3934
TOptionItem Class 3934
TOptionItem Members 3934
TOptionItem Methods 3935
TOptionItem Properties 3936

TOptionItems Class 3938
TOptionItems Members 3938
TOptionItems Methods 3938
TOptionItems Properties 3939
TRibbonActionPopupMenu Class 3940
TRibbonActionPopupMenu Members 3941
TRibbonActionPopupMenu Methods 3941
TRibbonActionPopupMenu Properties 3942
TRibbonApplicationMenuBar Class 3942
TRibbonApplicationMenuBar Members 3943
TRibbonApplicationMenuBar.Create Constructor 3944
TRibbonApplicationMenuBar.Destroy Destructor 3944
TRibbonApplicationMenuBar Methods 3945
TRibbonApplicationMenuBar Properties 3949
TRibbonApplicationPopupMenu Class 3961
TRibbonApplicationPopupMenu Members 3961
TRibbonApplicationPopupMenu.Create Constructor 3962
TRibbonApplicationPopupMenu.Destroy Destructor 3962
TRibbonApplicationPopupMenu Methods 3962
TRibbonQATCustomizeMenu Class 3968
TRibbonQATCustomizeMenu Members 3968
TRibbonQATCustomizeMenu Methods 3968
TRibbonRecentDocuments Class 3970
TRibbonRecentDocuments Members 3971
TRibbonRecentDocuments.Create Constructor 3971
TRibbonRecentDocuments Methods 3972
TRibbonRecentDocuments Properties 3972
TRibbonStyleMainPopupMenu Class 3975
TRibbonStyleMainPopupMenu Members 3975
TRibbonStyleMainPopupMenu.Create Constructor 3976
TRibbonStyleMainPopupMenu Methods 3976
TRibbonStylePopupMenu Class 3977
TRibbonStylePopupMenu Members 3977
TRibbonStylePopupMenu.Create Constructor 3977
TRibbonStylePopupMenu.Destroy Destructor 3978
TRibbonStylePopupMenu Methods 3978
TRibbonStylePopupMenu Properties 3980

RibbonGalleryBar Namespace 3981

Classes **3982**
TCustomRibbonGalleryBar Class 3982
TCustomRibbonGalleryBar Members 3982
TCustomRibbonGalleryBar.Create Constructor 3983
TCustomRibbonGalleryBar Methods 3983
TCustomRibbonGalleryBar Properties 3984
TCustomRibbonGalleryPopupBar Class 3988
TCustomRibbonGalleryPopupBar Members 3989
TCustomRibbonGalleryPopupBar Methods 3989
TRibbonGalleryBar Class 3990
TRibbonGalleryBar Members 3991
TRibbonGalleryBar Properties 3991
TRibbonGalleryPopupBar Class 3992
TRibbonPopupGalleryBar Class 3992
TRibbonPopupGalleryBar Members 3993
TRibbonPopupGalleryBar Methods 3993

RibbonLunaStyleActnCtrls Namespace 3997

Classes **3998**
TRibbonLunaSkin Class 3998
TRibbonLunaSkin Members 3998
TRibbonLunaSkin Methods 3999
TRibbonLunaStyleActionBars Class 3999
TRibbonLunaStyleActionBars Members 4000
TRibbonLunaStyleActionBars Methods 4000

Variables **4002**
RibbonLunaStyleActnCtrls.RibbonLunaStyle Variable 4002

RibbonObsidianStyleActnCtrls Namespace 4003

Classes **4004**
TRibbonObsidianSkin Class 4004
TRibbonObsidianSkin Members 4004
TRibbonObsidianSkin Methods 4005
TRibbonObsidianStyleActionBars Class 4005
TRibbonObsidianStyleActionBars Members 4006
TRibbonObsidianStyleActionBars Methods 4006

Variables **4008**

RibbonObsidianStyleActnCtrls.RibbonObsidianStyle Variable 4008

RibbonSilverStyleActnCtrls Namespace 4009

Classes **4010**

TRibbonSilverSkin Class 4010

TRibbonSilverSkin Members 4010

TRibbonSilverSkin Methods 4011

TRibbonSilverStyleActionBars Class 4011

TRibbonSilverStyleActionBars Members 4012

TRibbonSilverStyleActionBars Methods 4012

Variables **4014**

RibbonSilverStyleActnCtrls.RibbonSilverStyle Variable 4014

RibbonStyleActnCtrls Namespace 4015

Classes **4017**

TAlphaBitmap Class 4017

TAlphaBitmap Members 4017

TAlphaBitmap.Create Constructor 4018

TCustomRibbonSkin Class 4018

TCustomRibbonSkin Members 4018

TCustomRibbonSkin.Create Constructor 4019

TCustomRibbonSkin.Destroy Destructor 4019

TCustomRibbonSkin Enumerations 4019

TCustomRibbonSkin Methods 4020

TCustomRibbonSkin Properties 4022

TRibbonStyleActionBars Class 4022

TRibbonStyleActionBars Members 4023

TRibbonStyleActionBars.Create Constructor 4023

TRibbonStyleActionBars.Destroy Destructor 4023

TRibbonStyleActionBars Methods 4024

TRibbonStyleActionBars Properties 4028

Structs, Records, Enums **4029**

RibbonStyleActnCtrls.TRibbonElement Enumeration 4029

RibbonStyleActnCtrls.TSkinApplicationMenu Enumeration 4030

RibbonStyleActnCtrls.TSkinButton Enumeration 4032

RibbonStyleActnCtrls.TSkinCheckBox Enumeration 4036

RibbonStyleActnCtrls.TSkinComboBox Enumeration 4037

RibbonStyleActnCtrls.TSkinForm Enumeration 4038
RibbonStyleActnCtrls.TSkinGallery Enumeration 4040
RibbonStyleActnCtrls.TSkinMenu Enumeration 4041
RibbonStyleActnCtrls.TSkinQuickAccessToolbar Enumeration 4043
RibbonStyleActnCtrls.TSkinRadioButton Enumeration 4045
RibbonStyleActnCtrls.TSkinRibbon Enumeration 4045
RibbonStyleActnCtrls.TSkinRibbonGroup Enumeration 4046
RibbonStyleActnCtrls.TSkinScrollBar Enumeration 4048
RibbonStyleActnCtrls.TSkinSpinEdit Enumeration 4049
RibbonStyleActnCtrls.TSkinStatusBar Enumeration 4050
RibbonStyleActnCtrls.TSkinTab Enumeration 4051

Types **4053**
RibbonStyleActnCtrls.TCustomRibbonSkinClass Type 4053

Rio Namespace 4055

Classes **4056**
TRIO Class 4056
TRIO Members 4057
TRIO.Create Constructor 4058
TRIO.Destroy Destructor 4058
TRIO Fields 4058
TRIO Methods 4060
TRIO Properties 4063
TRIO Events 4065

Interfaces **4068**
IRIOAccess Interface 4068
IRIOAccess Members 4068
IRIOAccess Methods 4069
IRIOAccess Properties 4069

Types **4070**
Rio.TAfterExecuteEvent Type 4070
Rio.TBeforeExecuteEvent Type 4070

Constants **4072**
Rio.StubSize Constant 4072

Index a

1 IB Namespace

This is namespace IB.

Topics

Name	Description
Classes (see page 3)	The following table lists classes in this documentation.
Functions (see page 7)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 11)	The following table lists structs, records, enums in this documentation.
Types (see page 17)	The following table lists types in this documentation.
Variables (see page 20)	The following table lists variables in this documentation.
Constants (see page 21)	The following table lists constants in this documentation.

Classes

	Name	Description
	EIBClientError (see page 3)	EIBClientError represents a client-specific error in an InterBase Express application.
	EIBError (see page 3)	EIBError is the exception class for InterBase errors.
	EIBInterBaseError (see page 5)	EIBInterBaseError is used to raise server-specific errors.
	EIBInterBaseRoleError (see page 6)	EIBInterbaseRoleError is the exception class for InterBase errors that arise when a caller's role does not have sufficient rights.

Constants

Name	Description
IBBigLocalBufferLength (see page 21)	This is constant IB.IBBigLocalBufferLength.
IBErrorMessages (see page 21)	This is constant IB.IBErrorMessages.
IBHugeLocalBufferLength (see page 22)	This is constant IB.IBHugeLocalBufferLength.
IBLocalBufferLength (see page 23)	This is constant IB.IBLocalBufferLength.
IBPalette1 (see page 23)	This is constant IB.IBPalette1.
IBPalette2 (see page 23)	This is constant IB.IBPalette2.

Functions

	Name	Description
	CheckStatusVector (see page 7)	Sets or unsets status vector checking.
	GetIBDataBaseErrorMessages (see page 7)	Returns the current error message setting.
	IBAlloc (see page 8)	Allocates or de-allocates memory for a given size, and initializes the new memory to zero.

	IBDataBaseError (see page 8)	Interprets the SQLCode and IBErrorCode status and passes it to the EIBInterBaseError (see page 5) exception.
	IBError (see page 8)	Returns the error message for the specified error code.
	SetIBDataBaseErrorMessages (see page 9)	Enables you to choose the error messages you want to see.
	StatusVector (see page 9)	Retrieves the thread-specific status vector from thread-local storage.
	StatusVectorArray (see page 9)	Returns a pointer to the status vector array.
	StatusVectorAsText (see page 10)	Gets the status vector and returns it as a formatted string.

Structs, Records, Enums

	Name	Description
	TIBClientError (see page 11)	This is record IB.TIBClientError.
	TIBDataBaseErrorMessage (see page 15)	This is record IB.TIBDataBaseErrorMessage.
	TTraceFlag (see page 15)	This is record IB.TTraceFlag.

Types

Name	Description
PStatusVector (see page 17)	This is type IB.PStatusVector.
TIBDataBaseErrorMessages (see page 17)	TIBDataBaseErrorMessage (see page 15) and TIBDataBaseErrorMessages indicate the various strings available to describe an error that an IBX component encountered.
TStatusVector (see page 18)	This is type IB.TStatusVector.
TTraceFlags (see page 18)	TTraceFlag (see page 15) and TTraceFlags indicate the types of messages that pass between an application and the InterBase server.

Variables

Name	Description
IBCS (see page 20)	This is variable IB.IBCS.

1.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	EIBClientError (see page 3)	EIBClientError represents a client-specific error in an InterBase Express application.
	EIBError (see page 3)	EIBError is the exception class for InterBase errors.
	EIBInterBaseError (see page 5)	EIBInterBaseError is used to raise server-specific errors.
	EIBInterBaseRoleError (see page 6)	EIBInterbaseRoleError is the exception class for InterBase errors that arise when a caller's role does not have sufficient rights.

1.1.1 EIBClientError Class

EIBClientError represents a client-specific error in an InterBase Express application.

Class Hierarchy

Pascal

```
EIBClientError = class(EIBError);
```

C++

```
class EIBClientError : public EIBError;
```

File

IB

Description

Use EIBClientError to raise an exception for client-specific errors. It is raised when an application tries to use a component improperly. For example, an EIBClientError is raised when a component can't access the database because it does not have an IBDatabase (see page 163) component specified, attempts to edit a read-only dataset, tries to execute an empty query, and so on.

1.1.2 EIBError Class

EIBError is the exception class for InterBase errors.

Class Hierarchy

Pascal

```
EIBError = class(EDatabaseError);
```

C++

```
class EIBError : public EDatabaseError;
```

File

IB

Description

Use EIBError to raise an exception when a component detects an error in the database or in the component implementation. EIBError is the base class for all errors that arise in the use of InterBase Express. Its descendants represent specific types of errors, and include EIBClientError (see page 3), EIBInterBaseError (see page 5), and EIBInterBaseRoleError (see page 6).

1.1.2.1 EIBError Members

The following tables list the members exposed by EIBError.

Public Methods

	Name	Description
	Create (see page 4)	This is the overview for the Create constructor overload.

Public Properties

	Name	Description
R	IBErrorCode (see page 5)	Returns the InterBase error code.
R	SQLCode (see page 5)	Translates an InterBase error code in the error status vector to an SQL error number code.

1.1.2.2 Create Constructor

This is the overview for the Create constructor overload.

Overload List

	Name	Description
	EIBError.Create (Long, Long, string) (see page 4)	Calls the inherited Create and sets the SQLCode (see page 5) and IBErrorCode (see page 5)
	EIBError.Create (Long, string) (see page 4)	Calls the inherited Create and sets the SQLCode (see page 5) and IBErrorCode (see page 5)

1.1.2.2.1 EIBError.Create Constructor (Long, Long, string)

Calls the inherited Create and sets the SQLCode (see page 5) and IBErrorCode (see page 5)

Pascal

```
constructor Create(ASQLCode: Long; Msg: string); overload;
constructor Create(ASQLCode: Long; AIBErrorCode: Long; Msg: string); overload;
```

C++

```
__fastcall EIBError(Long ASQLCode, UnicodeString Msg);
__fastcall EIBError(Long ASQLCode, Long AIBErrorCode, UnicodeString Msg);
```

Description

Note: Use Create to call to the inherited Create and sets the SQL code and IBErrorCode (see page 5).

1.1.2.3 EIBError Properties

The properties of the EIBError class are listed here.

Public Properties

	Name	Description
	IBErrorCode (see page 5)	Returns the InterBase error code.
	SQLCode (see page 5)	Translates an InterBase error code in the error status vector to an SQL error number code.

1.1.2.3.1 EIBError.IBErrorCode Property

Returns the InterBase error code.

Pascal

```
property IBErrorCode: Long;
```

C++

```
__property Long IBErrorCode;
```

Description

Use IBErrorCode to get the InterBase error code.

1.1.2.3.2 EIBError.SQLCode Property

Translates an InterBase error code in the error status vector to an SQL error number code.

Pascal

```
property SQLCode: Long;
```

C++

```
__property Long SQLCode;
```

Description

Translates an InterBase error code in the error status vector to an SQL error number code.

Delphi syntax:

```
property SQLCode: Long;
```

C++ syntax:

```
__property
        int SQLCode = {read=FSQLCode, nodefault};
```

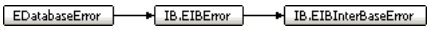
Description

Use SQLCode to translate an InterBase error code in the error status vector to an SQL error number code. Typically, this call is used to populate a program variable with an SQL error number for use in an SQL error-handling routine.

1.1.3 EIBInterBaseError Class

EIBInterBaseError is used to raise server-specific errors.

Class Hierarchy



Pascal

```
EIBInterBaseError = class(EIBError);
```

C++

```
class EIBInterBaseError : public EIBError;
```

File

IB

Description

Use EIBInterBaseError to raise an exception that represents an error code that the InterBase server returns.

1.1.4 EIBInterBaseRoleError Class

EIBInterbaseRoleError is the exception class for InterBase errors that arise when a caller's role does not have sufficient rights.

Class Hierarchy

Pascal

```
EIBInterBaseRoleError = class(EIBError);
```

C++

```
class EIBInterBaseRoleError : public EIBError;
```

File

IB

Description

EIBInterbaseRoleError is raised when InterBase reports failure due to an incorrect role.

1.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	CheckStatusVector (see page 7)	Sets or unsets status vector checking.
	GetIBDataBaseErrorMessages (see page 7)	Returns the current error message setting.
	IBAlloc (see page 8)	Allocates or de-allocates memory for a given size, and initializes the new memory to zero.
	IBDataBaseError (see page 8)	Interprets the SQLCode and IBErrorCode status and passes it to the EIBInterBaseError (see page 5) exception.
	IBError (see page 8)	Returns the error message for the specified error code.
	SetIBDataBaseErrorMessages (see page 9)	Enables you to choose the error messages you want to see.
	StatusVector (see page 9)	Retrieves the thread-specific status vector from thread-local storage.
	StatusVectorArray (see page 9)	Returns a pointer to the status vector array.
	StatusVectorAsText (see page 10)	Gets the status vector and returns it as a formatted string.

1.2.1 IB.CheckStatusVector Function

Sets or unsets status vector checking.

Pascal

```
function CheckStatusVector(ErrorCodes: array of ISC_STATUS): Boolean;
```

C++

```
Boolean CheckStatusVector(array of ISC_STATUS ErrorCodes);
```

File

IB

Description

Set CheckStatusVector to true to enable status vector checking.

1.2.2 IB.GetIBDataBaseErrorMessages Function

Returns the current error message setting.

Pascal

```
function GetIBDataBaseErrorMessages: TIBDataBaseErrorMessages;
```

C++

```
TIBDataBaseErrorMessages GetIBDataBaseErrorMessages();
```

File

IB

Description

Use the GetIBDataBaseErrorMessages function to return the current error message setting.

1.2.3 IB.IBAlloc Function

Allocates or de-allocates memory for a given size, and initializes the new memory to zero.

Pascal

```
procedure IBAlloc(var P; OldSize: Integer; NewSize: Integer);
```

C++

```
IBAlloc( P, int OldSize, int NewSize);
```

File

IB

Description

Use IBAlloc to allocate or de-allocate memory for a given size, and initialize the new memory to zero.

1.2.4 IB.IBDataBaseError Function

Interprets the SQLCode and IBErrorCode status and passes it to the EIBInterBaseError (see page 5) exception.

Pascal

```
procedure IBDataBaseError;
```

C++

```
IBDataBaseError();
```

File

IB

Description

Use the IBDataBaseError procedure to look at the status vector, interpret the SQLCode and IBErrorCode status, construct the status string, and pass it to the EIBInterBaseError (see page 5) exception.

1.2.5 IB.IBError Function

Returns the error message for the specified error code.

Pascal

```
procedure IBError(ErrMess: TIBClientError; const Args: array of const);
```

C++

```
IBError(TIBClientError ErrMess, const array of const Args);
```

File

IB

Description

Use the IBError procedure to look up the appropriate error message for a specified error code and raise an EIBClientError (see page 3) exception.

1.2.6 IB.SetIBDataBaseErrorMessages Function

Enables you to choose the error messages you want to see.

Pascal

```
procedure SetIBDataBaseErrorMessages(Value: TIBDataBaseErrorMessages);
```

C++

```
SetIBDataBaseErrorMessages(TIBDataBaseErrorMessages Value);
```

File

IB

Description

Use the SetIBDataBaseErrorMessages procedure to choose the error messages that you want to use. The available options are defined in TIBDataBaseErrorMessages (see page 17).

1.2.7 IB.StatusVector Function

Retrieves the thread-specific status vector from thread-local storage.

Pascal

```
function StatusVector: PISC_STATUS;
```

C++

```
PISC_STATUS StatusVector();
```

File

IB

Description

Use the StatusVector function to retrieve the thread-specific status from the thread-local storage.

1.2.8 IB.StatusVectorArray Function

Returns a pointer to the status vector array.

Pascal

```
function StatusVectorArray: PStatusVector;
```

C++

```
PStatusVector StatusVectorArray();
```

File

IB

Description

Use the StatusVectorArray function to return a pointer to the status vector array from the thread-local storage.

1.2.9 IB.StatusVectorAsText Function

Gets the status vector and returns it as a formatted string.

Pascal

```
function StatusVectorAsText: string;
```

C++

```
UnicodeString StatusVectorAsText();
```

File

IB

Description

Use the StatusVector (see page 9) function to retrieve the status vector and return it as a formatted string.

1.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TIBClientError (see page 11)	This is record IB.TIBClientError.
	TIBDataBaseErrorMessage (see page 15)	This is record IB.TIBDataBaseErrorMessage.
	TTraceFlag (see page 15)	This is record IB.TTraceFlag.

1.3.1 IB.TIBClientError Enumeration

Pascal

```
TIBClientError = (
  ibxeUnknownError,
  ibxeInterBaseMissing,
  ibxeInterBaseInstallMissing,
  ibxeIB60feature,
  ibxeNotSupported,
  ibxeNotPermitted,
  ibxeFileAccessError,
  ibxeConnectionTimeout,
  ibxeCannotSetDatabase,
  ibxeCannotSetTransaction,
  ibxeOperationCancelled,
  ibxeDPBConstantNotSupported,
  ibxeDPBConstantUnknown,
  ibxeTPBConstantNotSupported,
  ibxeTPBConstantUnknown,
  ibxeDatabaseClosed,
  ibxeDatabaseOpen,
  ibxeDatabaseNameMissing,
  ibxeNotInTransaction,
  ibxeInTransaction,
  ibxeTimeoutNegative,
  ibxeNoDatabasesInTransaction,
  ibxeUpdateWrongDB,
  ibxeUpdateWrongTR,
  ibxeDatabaseNotAssigned,
  ibxeTransactionNotAssigned,
  ibxeXSQLDAIndexOutOfRange,
  ibxeXSQLDANameDoesNotExist,
  ibxeEOF,
  ibxeBOF,
  ibxeInvalidStatementHandle,
  ibxeSQLOpen,
  ibxeSQLClosed,
  ibxeDatasetOpen,
  ibxeDatasetClosed,
  ibxeUnknownSQLDataType,
  ibxeInvalidColumnIndex,
  ibxeInvalidParamColumnIndex,
  ibxeInvalidDataConversion,
  ibxeColumnIsNotNullable,
  ibxeBlobCannotBeRead,
```

```
    ibxeBlobCannotBeWritten,
    ibxeEmptyQuery,
    ibxeCannotOpenNonSQLSelect,
    ibxeNoFieldAccess,
    ibxeFieldReadOnly,
    ibxeFieldNotFound,
    ibxeNotEditing,
    ibxeCannotInsert,
    ibxeCannotPost,
    ibxeCannotUpdate,
    ibxeCannotDelete,
    ibxeCannotRefresh,
    ibxeBufferNotSet,
    ibxeCircularReference,
    ibxeSQLParseError,
    ibxeUserAbort,
    ibxeDataSetUniDirectional,
    ibxeCannotCreateSharedResource,
    ibxeWindowsAPIError,
    ibxeColumnListsDontMatch,
    ibxeColumnTypesDontMatch,
    ibxeCantEndSharedTransaction,
    ibxeFieldUnsupportedType,
    ibxeCircularDataLink,
    ibxeEmptySQLStatement,
    ibxeIsASelectStatement,
    ibxeRequiredParamNotSet,
    ibxeNoStoredProcName,
    ibxeIsAExecuteProcedure,
    ibxeUpdateFailed,
    ibxeNotCachedUpdates,
    ibxeNotLiveRequest,
    ibxeNoProvider,
    ibxeNoRecordsAffected,
    ibxeNoTableName,
    ibxeCannotCreatePrimaryIndex,
    ibxeCannotDropSystemIndex,
    ibxeTableNameMismatch,
    ibxeIndexFieldMissing,
    ibxeInvalidCancellation,
    ibxeInvalidEvent,
    ibxeMaximumEvents,
    ibxeNoEventsRegistered,
    ibxeInvalidQueueing,
    ibxeInvalidRegistration,
    ibxeInvalidBatchMove,
    ibxeSQLDialectInvalid,
    ibxeSPBConstantNotSupported,
    ibxeSPBConstantUnknown,
    ibxeServiceActive,
    ibxeServiceInActive,
    ibxeServerNameMissing,
    ibxeQueryParamsError,
    ibxeStartParamsError,
    ibxeOutputParsingError,
    ibxeUseSpecificProcedures,
    ibxeSQLMonitorAlreadyPresent,
    ibxeCantPrintValue,
    ibxeEOFReached,
    ibxeEOFInComment,
    ibxeEOFInString,
    ibxeParamNameExpected,
    ibxeSuccess,
    ibxeDelphiException,
    ibxeNoOptionsSet,
    ibxeNoDestinationDirectory,
```

```
    ibxeNosourceDirectory,
    ibxeNoUninstallFile,
    ibxeOptionNeedsClient,
    ibxeOptionNeedsServer,
    ibxeInvalidOption,
    ibxeInvalidOnErrorResult,
    ibxeInvalidOnStatusResult,
    ibxeDPBConstantUnknownEx,
    ibxeTPBConstantUnknownEx,
    ibxeUnknownPlan,
    ibxeFieldSizeMismatch,
    ibxeEventAlreadyRegistered,
    ibxeStringTooLarge
  );
```

C++

```
  enum TIBClientError {
    ibxeUnknownError,
    ibxeInterBaseMissing,
    ibxeInterBaseInstallMissing,
    ibxeIB60feature,
    ibxeNotSupported,
    ibxeNotPermitted,
    ibxeFileAccessError,
    ibxeConnectionTimeout,
    ibxeCannotSetDatabase,
    ibxeCannotSetTransaction,
    ibxeOperationCancelled,
    ibxeDPBConstantNotSupported,
    ibxeDPBConstantUnknown,
    ibxeTPBConstantNotSupported,
    ibxeTPBConstantUnknown,
    ibxeDatabaseClosed,
    ibxeDatabaseOpen,
    ibxeDatabaseNameMissing,
    ibxeNotInTransaction,
    ibxeInTransaction,
    ibxeTimeoutNegative,
    ibxeNoDatabasesInTransaction,
    ibxeUpdateWrongDB,
    ibxeUpdateWrongTR,
    ibxeDatabaseNotAssigned,
    ibxeTransactionNotAssigned,
    ibxeXSQLDAIndexOutOfRange,
    ibxeXSQLDANameDoesNotExist,
    ibxeEOF,
    ibxeBOF,
    ibxeInvalidStatementHandle,
    ibxeSQLOpen,
    ibxeSQLClosed,
    ibxeDatasetOpen,
    ibxeDatasetClosed,
    ibxeUnknownSQLDataType,
    ibxeInvalidColumnIndex,
    ibxeInvalidParamColumnIndex,
    ibxeInvalidDataConversion,
    ibxeColumnIsNotNullable,
    ibxeBlobCannotBeRead,
    ibxeBlobCannotBeWritten,
    ibxeEmptyQuery,
    ibxeCannotOpenNonSQLSelect,
    ibxeNoFieldAccess,
    ibxeFieldReadOnly,
    ibxeFieldNotFound,
    ibxeNotEditing,
    ibxeCannotInsert,
```

```
    ibxeCannotPost,
    ibxeCannotUpdate,
    ibxeCannotDelete,
    ibxeCannotRefresh,
    ibxeBufferNotSet,
    ibxeCircularReference,
    ibxeSQLParseError,
    ibxeUserAbort,
    ibxeDataSetUniDirectional,
    ibxeCannotCreateSharedResource,
    ibxeWindowsAPIError,
    ibxeColumnListsDontMatch,
    ibxeColumnTypesDontMatch,
    ibxeCantEndSharedTransaction,
    ibxeFieldUnsupportedType,
    ibxeCircularDataLink,
    ibxeEmptySQLStatement,
    ibxeIsASelectStatement,
    ibxeRequiredParamNotSet,
    ibxeNoStoredProcName,
    ibxeIsAExecuteProcedure,
    ibxeUpdateFailed,
    ibxeNotCachedUpdates,
    ibxeNotLiveRequest,
    ibxeNoProvider,
    ibxeNoRecordsAffected,
    ibxeNoTableName,
    ibxeCannotCreatePrimaryIndex,
    ibxeCannotDropSystemIndex,
    ibxeTableNameMismatch,
    ibxeIndexFieldMissing,
    ibxeInvalidCancellation,
    ibxeInvalidEvent,
    ibxeMaximumEvents,
    ibxeNoEventsRegistered,
    ibxeInvalidQueueing,
    ibxeInvalidRegistration,
    ibxeInvalidBatchMove,
    ibxeSQLDialectInvalid,
    ibxeSPBConstantNotSupported,
    ibxeSPBConstantUnknown,
    ibxeServiceActive,
    ibxeServiceInActive,
    ibxeServerNameMissing,
    ibxeQueryParamsError,
    ibxeStartParamsError,
    ibxeOutputParsingError,
    ibxeUseSpecificProcedures,
    ibxeSQLMonitorAlreadyPresent,
    ibxeCantPrintValue,
    ibxeEOFReached,
    ibxeEOFInComment,
    ibxeEOFInString,
    ibxeParamNameExpected,
    ibxeSuccess,
    ibxeDelphiException,
    ibxeNoOptionsSet,
    ibxeNoDestinationDirectory,
    ibxeNosourceDirectory,
    ibxeNoUninstallFile,
    ibxeOptionNeedsClient,
    ibxeOptionNeedsServer,
    ibxeInvalidOption,
    ibxeInvalidOnErrorResult,
    ibxeInvalidOnStatusResult,
    ibxeDPBConstantUnknownEx,
```

```
    ibxeTPBConstantUnknownEx,
    ibxeUnknownPlan,
    ibxeFieldSizeMismatch,
    ibxeEventAlreadyRegistered,
    ibxeStringTooLarge
  };
```

File

IB

Description

This is record IB.TIBClientError.

1.3.2 IB.TIBDataBaseErrorMessage Enumeration

Pascal

```
  TIBDataBaseErrorMessage = (
    ShowSQLCode,
    ShowIBMessage,
    ShowSQLMessage
  );
```

C++

```
  enum TIBDataBaseErrorMessage {
    ShowSQLCode,
    ShowIBMessage,
    ShowSQLMessage
  };
```

File

IB

Description

This is record IB.TIBDataBaseErrorMessage.

1.3.3 IB.TTraceFlag Enumeration

Pascal

```
  TTraceFlag = (
    tfQPrepare,
    tfQExecute,
    tfQFetch,
    tfError,
    tfStmt,
    tfConnect,
    tfTransact,
    tfBlob,
    tfService,
    tfMisc
  );
```

C++

```
  enum TTraceFlag {
    tfQPrepare,
```

```
    tfQExecute,
    tfQFetch,
    tfError,
    tfStmt,
    tfConnect,
    tfTransact,
    tfBlob,
    tfService,
    tfMisc
  };
```

File

IB

Description

This is record IB.TTraceFlag.

1.4 Types

The following table lists types in this documentation.

Types

Name	Description
PStatusVector (see page 17)	This is type IB.PStatusVector.
TIBDataBaseErrorMessages (see page 17)	TIBDataBaseErrorMessage (see page 15) and TIBDataBaseErrorMessages indicate the various strings available to describe an error that an IBX component encountered.
TStatusVector (see page 18)	This is type IB.TStatusVector.
TTraceFlags (see page 18)	TTraceFlag (see page 15) and TTraceFlags indicate the types of messages that pass between an application and the InterBase server.

1.4.1 IB.PStatusVector Type

Pascal

```
PStatusVector = ^TStatusVector;
```

C++

```
^TStatusVector PStatusVector;
```

File

IB

Description

This is type IB.PStatusVector.

1.4.2 IB.TIBDataBaseErrorMessages Type

TIBDataBaseErrorMessage (see page 15) and TIBDataBaseErrorMessages indicate the various strings available to describe an error that an IBX component encountered.

Pascal

```
TIBDataBaseErrorMessages = set of TIBDataBaseErrorMessage;
```

C++

```
TIBDataBaseErrorMessage TIBDataBaseErrorMessages;
```

File

IB

Description

TIBDataBaseErrorMessages is a set of TIBDataBaseErrorMessage (see page 15) values. The following table lists the possible values:

Value Meaning

ShowSQLCode The SQL code that the server was trying to execute when the error occurred.

ShowIBMessage The error message InterBase provides for the error.

ShowSQLMessage The error message that describes what part of the SQL code caused the error.

1.4.3 IB.TStatusVector Type

Pascal

```
TStatusVector = array[0..19] of ISC_STATUS;
```

C++

```
array[0..19] of ISC_STATUS TStatusVector;
```

File

IB

Description

This is type IB.TStatusVector.

1.4.4 IB.TTraceFlags Type

TTraceFlag (see page 15) and TTraceFlags indicate the types of messages that pass between an application and the InterBase server.

Pascal

```
TTraceFlags = set of TTraceFlag;
```

C++

```
TTraceFlag TTraceFlags;
```

File

IB

Description

TTraceFlags is a set of TTraceFlag (see page 15) values. The following table lists the possible values:

Values Meaning

tfQPrepare Monitor Prepare statements.

tfQExecute Monitor ExecSQL statements.

tfError Monitor server error messages. Such messages may include an error code.

tfStmt Monitor all SQL statements.

tfConnect Monitor database connect and disconnect operations, including allocation of connection handles, and freeing connection handles.

tfTransact Monitor transaction statements, such as StartTransaction, Commit, and Rollback.

tfBlob Monitor operations on blob data types.

tfService Monitor messages to one of the IB (see page 1) services.

tfMisc Monitor any statements not covered by other flag options.

1.5 Variables

The following table lists variables in this documentation.

Variables

Name	Description
IBCS (see page 20)	This is variable IB.IBCS.

1.5.1 IB.IBCS Variable

Pascal

```
IBCS: TRTLCriticalSection;
```

C++

```
TRTLCriticalSection IBCS;
```

File

IB

Description

This is variable IB.IBCS.

1.6 Constants

The following table lists constants in this documentation.

Constants

Name	Description
IBBigLocalBufferLength (see page 21)	This is constant IB.IBBigLocalBufferLength.
IBErrorMessages (see page 21)	This is constant IB.IBErrorMessages.
IBHugeLocalBufferLength (see page 22)	This is constant IB.IBHugeLocalBufferLength.
IBLocalBufferLength (see page 23)	This is constant IB.IBLocalBufferLength.
IBPalette1 (see page 23)	This is constant IB.IBPalette1.
IBPalette2 (see page 23)	This is constant IB.IBPalette2.

1.6.1 IB.IBBigLocalBufferLength Constant

Pascal

```
IBBigLocalBufferLength = IBLocalBufferLength * 2;
```

C++

```
IBBigLocalBufferLength = IBLocalBufferLength * 2;
```

File

IB

Description

This is constant IB.IBBigLocalBufferLength.

1.6.2 IB.IBErrorMessages Constant

Pascal

```
IBErrorMessages: array[TIBClientError] of string = ( SUnknownError, SInterBaseMissing,
SInterBaseInstallMissing, SIB60feature, SNotSupported, SNotPermitted, SFileAccessError,
SConnectionTimeout, SCannotSetDatabase, SCannotSetTransaction, SOperationCancelled,
SDPBConstantNotSupported, SDPBConstantUnknown, STPBConstantNotSupported, STPBConstantUnknown,
SDatabaseClosed, SDatabaseOpen, SDatabaseNameMissing, SNotInTransaction, SInTransaction,
STimeoutNegative, SNoDatabasesInTransaction, SUpdateWrongDB, SUpdateWrongTR,
SDatabaseNotAssigned, STransactionNotAssigned, SXSQLDAIndexOutOfRange,
SXSQLDANameDoesNotExist, SEOF, SBOF, SInvalidStatementHandle, SSQLOpen, SSQLClosed,
SDatasetOpen, SDatasetClosed, SUnknownSQLDataType, SInvalidColumnIndex,
SInvalidParamColumnIndex, SInvalidDataConversion, SColumnIsNotNullable, SBlobCannotBeRead,
SBlobCannotBeWritten, SEmptyQuery, SCannotOpenNonSQLSelect, SNoFieldAccess, SFieldReadOnly,
SFieldNotFound, SNotEditing, SCannotInsert, SCannotPost, SCannotUpdate, SCannotDelete,
SCannotRefresh, SBufferNotSet, SCircularReference, SSQLParseError, SUserAbort,
SDataSetUniDirectional, SCannotCreateSharedResource, SWindowsAPIError, SColumnListsDontMatch,
SColumnTypesDontMatch, SCantEndSharedTransaction, SFieldUnsupportedType, SCircularDataLink,
SEmptySQLStatement, SIsASelectStatement, SRequiredParamNotSet, SNoStoredProcName,
SIsAExecuteProcedure, SUpdateFailed, SNotCachedUpdates, SNotLiveRequest, SNoProvider,
SNoRecordsAffected, SNoTableName, SCannotCreatePrimaryIndex, SCannotDropSystemIndex,
```

```
  STableNameMismatch, SIndexFieldMissing, SInvalidCancellation, SInvalidEvent, SMaximumEvents,
  SNoEventsRegistered, SInvalidQueueing, SInvalidRegistration, SInvalidBatchMove,
  SSQLDialectInvalid, SSPBConstantNotSupported, SSPBConstantUnknown, SServiceActive,
  SServiceInActive, SServerNameMissing, SQueryParamsError, SStartParamsError,
  SOutputParsingError, SUseSpecificProcedures, SSQLMonitorAlreadyPresent, SCantPrintValue,
  SEOFReached, SEOFInComment, SEOFInString, SParamNameExpected, SSuccess, SDelphiException,
  SNoOptionsSet, SNoDestinationDirectory, SNosourceDirectory, SNoUninstallFile,
  SOptionNeedsClient, SOptionNeedsServer, SInvalidOption, SInvalidOnErrorResult,
  SInvalidOnStatusResult, SDPBConstantUnknownEx, STPBConstantUnknownEx, SUnknownPlan,
  SFieldSizeMismatch, SEventAlreadyRegistered, SStringTooLarge );
```

C++

```
  array[TIBClientError] of string IBErrorMessages = ( SUnknownError, SInterBaseMissing,
  SInterBaseInstallMissing, SIB60feature, SNotSupported, SNotPermitted, SFileAccessError,
  SConnectionTimeout, SCannotSetDatabase, SCannotSetTransaction, SOperationCancelled,
  SDPBConstantNotSupported, SDPBConstantUnknown, STPBConstantNotSupported, STPBConstantUnknown,
  SDatabaseClosed, SDatabaseOpen, SDatabaseNameMissing, SNotInTransaction, SInTransaction,
  STimeoutNegative, SNoDatabasesInTransaction, SUpdateWrongDB, SUpdateWrongTR,
  SDatabaseNotAssigned, STransactionNotAssigned, SXSQLDAIndexOutOfRange,
  SXSQLDANameDoesNotExist, SEOF, SBOF, SInvalidStatementHandle, SSQLOpen, SSQLClosed,
  SDatasetOpen, SDatasetClosed, SUnknownSQLDataType, SInvalidColumnIndex,
  SInvalidParamColumnIndex, SInvalidDataConversion, SColumnIsNotNullable, SBlobCannotBeRead,
  SBlobCannotBeWritten, SEmptyQuery, SCannotOpenNonSQLSelect, SNoFieldAccess, SFieldReadOnly,
  SFieldNotFound, SNotEditing, SCannotInsert, SCannotPost, SCannotUpdate, SCannotDelete,
  SCannotRefresh, SBufferNotSet, SCircularReference, SSQLParseError, SUserAbort,
  SDataSetUniDirectional, SCannotCreateSharedResource, SWindowsAPIError, SColumnListsDontMatch,
  SColumnTypesDontMatch, SCantEndSharedTransaction, SFieldUnsupportedType, SCircularDataLink,
  SEmptySQLStatement, SIsASelectStatement, SRequiredParamNotSet, SNoStoredProcName,
  SIsAExecuteProcedure, SUpdateFailed, SNotCachedUpdates, SNotLiveRequest, SNoProvider,
  SNoRecordsAffected, SNoTableName, SCannotCreatePrimaryIndex, SCannotDropSystemIndex,
  STableNameMismatch, SIndexFieldMissing, SInvalidCancellation, SInvalidEvent, SMaximumEvents,
  SNoEventsRegistered, SInvalidQueueing, SInvalidRegistration, SInvalidBatchMove,
  SSQLDialectInvalid, SSPBConstantNotSupported, SSPBConstantUnknown, SServiceActive,
  SServiceInActive, SServerNameMissing, SQueryParamsError, SStartParamsError,
  SOutputParsingError, SUseSpecificProcedures, SSQLMonitorAlreadyPresent, SCantPrintValue,
  SEOFReached, SEOFInComment, SEOFInString, SParamNameExpected, SSuccess, SDelphiException,
  SNoOptionsSet, SNoDestinationDirectory, SNosourceDirectory, SNoUninstallFile,
  SOptionNeedsClient, SOptionNeedsServer, SInvalidOption, SInvalidOnErrorResult,
  SInvalidOnStatusResult, SDPBConstantUnknownEx, STPBConstantUnknownEx, SUnknownPlan,
  SFieldSizeMismatch, SEventAlreadyRegistered, SStringTooLarge );
```

File

IB

Description

This is constant IB.IBErrorMessages.

1.6.3 IB.IBHugeLocalBufferLength Constant

Pascal

```
  IBHugeLocalBufferLength = IBBigLocalBufferLength * 20;
```

C++

```
  IBHugeLocalBufferLength = IBBigLocalBufferLength * 20;
```

File

IB

Description

This is constant IB.IBHugeLocalBufferLength.

1.6.4 IB.IBLocalBufferLength Constant

Pascal

```
IBLocalBufferLength = 512;
```

C++

```
IBLocalBufferLength = 512;
```

File

IB

Description

This is constant IB.IBLocalBufferLength.

1.6.5 IB.IBPalette1 Constant

Pascal

```
IBPalette1 = 'InterBase';
```

C++

```
IBPalette1 = 'InterBase';
```

File

IB

Description

This is constant IB.IBPalette1.

1.6.6 IB.IBPalette2 Constant

Pascal

```
IBPalette2 = 'InterBase Admin';
```

C++

```
IBPalette2 = 'InterBase Admin';
```

File

IB

Description

This is constant IB.IBPalette2.

2 IBBatchMove Namespace

This is namespace IBBatchMove.

Topics

Name	Description
Classes (see page 26)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 38)	The following table lists structs, records, enums in this documentation.

Classes

	Name	Description
	TIBBatchMove (see page 26)	TIBBatchMove enables applications to perform database operations on groups of records or entire tables.

Structs, Records, Enums

	Name	Description
	TBatchMode (see page 38)	This is record IBBatchMove.TBatchMode.

2.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBBatchMove (see page 26)	TIBBatchMove enables applications to perform database operations on groups of records or entire tables.

2.1.1 TIBBatchMove Class

TIBBatchMove enables applications to perform database operations on groups of records or entire tables.

Class Hierarchy

Pascal

```
TIBBatchMove = class(TComponent);
```

C++

```
class TIBBatchMove : public TComponent;
```

File

IBBatchMove

Description

Set the properties of the TIBBatchMove object to

- Add the records in a dataset to a database table.
- Delete the records in a dataset from a database table.
- Copy a dataset to create a new database table or overwrite an existing table.

Call the Execute (see page 28) method to perform the actual operation. TIBTable (see page 600) objects can also perform the same batch operations, but only with the options specified by the default TIBBatchMove property values.

2.1.1.1 TIBBatchMove Members

The following tables list the members exposed by TIBBatchMove.

Protected Methods

	Name	Description
	Notification (see page 29)	Forwards notification messages to all owned components.

Public Methods

	Name	Description
	Create (see page 27)	Creates an instance of TIBBatchMove (see page 26).
	Destroy (see page 28)	Destroys an instance of TIBBatchMove (see page 26).

	Execute (see page 28)	Performs the batch operation specified by Mode (see page 34).

Public Properties

	Name	Description
R	ChangedCount (see page 31)	Indicates the number of records from the Destination (see page 32) table that are altered as a result of the batch operation.
R	KeyViolCount (see page 32)	Reports the number of records that could not be replaced, added, or deleted from Destination (see page 32) because of integrity or key violations.
R	MovedCount (see page 34)	Reports the number of records form the Source (see page 36) which were applied to the destination.
R	ProblemCount (see page 35)	Indicates the number of records which could not be added to Destination (see page 32) without loss of data due to a field type mismatch.

Published Properties

	Name	Description
	AbortOnKeyViol (see page 30)	Specifies whether the batch operation is terminated immediately when integrity or key violations occur.
	AbortOnProblem (see page 30)	Specifies whether the batch operation is terminated immediately when it is necessary to truncate data to make it fit in the specified Destination (see page 32).
	ChangedTableName (see page 31)	Specifies the name of a local Interbase table that is created to hold copies of all records from the destination table that are changed by the batch operation.
	CommitCount (see page 31)	Specifies how many records are batch moved before a commit occurs.
	Destination (see page 32)	Specifies the TIBTable (see page 600) object for the database table that is the destination of the batch operation.
	KeyViolTableName (see page 33)	Specifies the name of an InterBase table that will be created to contain all records from the Source (see page 36) that cannot participate in the batch operation because of integrity or key violations.
	Mappings (see page 33)	Specifies the column mappings for a batch operation.
	Mode (see page 34)	Specifies what the TIBBatchMove (see page 26) object does when the Execute (see page 28) method is called.
	ProblemTableName (see page 35)	Specifies the name of a Interbase table that will be created to contain all records from the Source (see page 36) that contain fields which were trimmed to match the corresponding field type in the Destination (see page 32) table.
	RecordCount (see page 36)	Specifies the maximum number of records that are applied to the Destination (see page 32) when Execute (see page 28) is called.
	Source (see page 36)	Specifies the dataset that is the source of the batch operation.
	Transliterate (see page 37)	Specifies whether the data in the Source (see page 36) records should be converted form the locale of the Source (see page 36) to the locale of the Destination (see page 32) when the Execute (see page 28) method is called.

2.1.1.2 TIBBatchMove.Create Constructor

Creates an instance of TIBBatchMove (see page 26).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBBatchMove(TComponent * AOwner);
```

Description

Call Create to create an instance of TIBBatchMove (see page 26) for performing a batch operation. Create allocates the memory for the TIBBatchMove (see page 26) object and

- Sets AbortOnKeyViol (see page 30) to True.
- Sets AbortOnProblem (see page 30) to True.
- Sets Transliterate (see page 37) to True.

- Creates a TStringList object to hold the Mappings (see page 33).

The property settings made by Create are the values used when a TIBTable (see page 600) object performs batch operations using its BatchMove method.

2.1.1.3 TIBBatchMove.Destroy Destructor

Destroys an instance of TIBBatchMove (see page 26).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBBatchMove();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TIBBatchMove (see page 26) object is not already freed and only then calls Destroy.

Before calling the inherited destructor, Destroy frees the TStringList object that implements the Mappings (see page 33) property.

2.1.1.4 TIBBatchMove Methods

The methods of the TIBBatchMove class are listed here.

Protected Methods

	Name	Description
	Notification (see page 29)	Forwards notification messages to all owned components.

Public Methods

	Name	Description
	Execute (see page 28)	Performs the batch operation specified by Mode (see page 34).

2.1.1.4.1 TIBBatchMove.Execute Method

Performs the batch operation specified by Mode (see page 34).

Pascal

```
procedure Execute;
```

C++

```
__fastcall Execute();
```

Description

After setting properties to indicate what batch operation should be performed and how, call Execute to perform the operation. As a minimum, the Mode (see page 34), Source (see page 36), and Destination (see page 32) properties must be set.

After calling Execute, the values of the ChangedCount (see page 31), KeyViolCount (see page 32), MovedCount (see page 34), ProblemCount (see page 35), and RecordCount (see page 36) properties will become available. These properties will give an indication of what happened as a result of the call to Execute. Additional information about the batch operation will be available through the creation of a changed table, a keyviol table, and/or a problems table if it was requested.

2.1.1.4.2 TIBBatchMove.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

2.1.1.5 TIBBatchMove Properties

The properties of the TIBBatchMove class are listed here.

Public Properties

	Name	Description
R	ChangedCount (see page 31)	Indicates the number of records from the Destination (see page 32) table that are altered as a result of the batch operation.
R	KeyViolCount (see page 32)	Reports the number of records that could not be replaced, added, or deleted from Destination (see page 32) because of integrity or key violations.
R	MovedCount (see page 34)	Reports the number of records form the Source (see page 36) which were applied to the destination.
R	ProblemCount (see page 35)	Indicates the number of records which could not be added to Destination (see page 32) without loss of data due to a field type mismatch.

Published Properties

	Name	Description
	AbortOnKeyViol (see page 30)	Specifies whether the batch operation is terminated immediately when integrity or key violations occur.
	AbortOnProblem (see page 30)	Specifies whether the batch operation is terminated immediately when it is necessary to truncate data to make it fit in the specified Destination (see page 32).
	ChangedTableName (see page 31)	Specifies the name of a local Interbase table that is created to hold copies of all records from the destination table that are changed by the batch operation.
	CommitCount (see page 31)	Specifies how many records are batch moved before a commit occurs.
	Destination (see page 32)	Specifies the TIBTable (see page 600) object for the database table that is the destination of the batch operation.

	KeyViolTableName (see page 33)	Specifies the name of an InterBase table that will be created to contain all records from the Source (see page 36) that cannot participate in the batch operation because of integrity or key violations.
	Mappings (see page 33)	Specifies the column mappings for a batch operation.
	Mode (see page 34)	Specifies what the TIBBatchMove (see page 26) object does when the Execute (see page 28) method is called.
	ProblemTableName (see page 35)	Specifies the name of a Interbase table that will be created to contain all records from the Source (see page 36) that contain fields which were trimmed to match the corresponding field type in the Destination (see page 32) table.
	RecordCount (see page 36)	Specifies the maximum number of records that are applied to the Destination (see page 32) when Execute (see page 28) is called.
	Source (see page 36)	Specifies the dataset that is the source of the batch operation.
	Transliterate (see page 37)	Specifies whether the data in the Source (see page 36) records should be converted form the locale of the Source (see page 36) to the locale of the Destination (see page 32) when the Execute (see page 28) method is called.

2.1.1.5.1 TIBBatchMove.AbortOnKeyViol Property

Specifies whether the batch operation is terminated immediately when integrity or key violations occur.

Pascal

```
property AbortOnKeyViol: Boolean;
```

C++

```
__property Boolean AbortOnKeyViol;
```

Description

Set AbortOnKeyViol to true (the default) to cancel the batch operation when it causes integrity or key violations in the destination table. Set AbortOnKeyViol to false to cause records that would cause integrity or key violations to be posted to a key violations table instead. Provide a KeyViolTableName (see page 33) when setting AbortOnKeyViol to false. If there is no KeyViolTableName (see page 33), AbortOnKeyViol may still be set to false, in which case problem records are counted but not saved in a table.

Key violations occur when appending records to a table that has a primary key. On a table with a primary key, no two records may have the same values on all of the key fields. If a record that is being appended has the same values on all the key fields as a record in the destination table, it cannot be appended.

Integrity violations occur when deleting, appending or updating records to a table that has a referential integrity specification. A referential integrity specification requires that certain fields in one table must have values that exist in a particular field of another table. If a record that is being added does not match the restrictions imposed by referential integrity, it cannot be added. If deleting record would cause tighter referential integrity restrictions that another table cannot meet, it cannot be deleted.

2.1.1.5.2 TIBBatchMove.AbortOnProblem Property

Specifies whether the batch operation is terminated immediately when it is necessary to truncate data to make it fit in the specified Destination (see page 32).

Pascal

```
property AbortOnProblem: Boolean;
```

C++

```
__property Boolean AbortOnProblem;
```

Description

Set AbortOnProblem to true to cancel the batch operation when types of the fields in the Source (see page 36) table records do not match the types of the fields in the Destination (see page 32) table records. Set AbortOnProblem to false to cause the

mismatched fields to be 'trimmed'. When AbortOnProblem is false, set the ProblemTableName (see page 35) to have all the untrimmed versions of records that were trimmed saved to a separate table.

Trimming a field means converting its value to a value compatible with the destination field type. For example, if a field in the destination table holds a string of size 10 characters, and the corresponding field in the source table holds a string of size 15 characters, any values from the source table longer than 10 characters must be truncated. Values that cannot be converted will cause TIBBatchMove (see page 26) to raise an exception.

Trimming does not occur when the Mode (see page 34) property is set to batCopy. Trimming fields on a delete operation means that records in the destination table that do not exactly match the values from the source table may be deleted.

2.1.1.5.3 TIBBatchMove.ChangedCount Property

Indicates the number of records from the Destination (see page 32) table that are altered as a result of the batch operation.

Pascal

```
property ChangedCount: Longint;
```

C++

```
__property Longint ChangedCount;
```

Description

Use ChangedCount to learn the number of records in the Destination (see page 32) table that are updated when the Mode (see page 34) is batUpdate or batAppendUpdate, or to learn the number that are deleted when the Mode (see page 34) is batDelete.

Set the ChangedTableName (see page 31) property to save the unchanged versions of the altered records to a separate table. ChangedCount is then the number of records in that separate table.

2.1.1.5.4 TIBBatchMove.ChangedTableName Property

Specifies the name of a local Interbase table that is created to hold copies of all records from the destination table that are changed by the batch operation.

Pascal

```
property ChangedTableName: TFileName;
```

C++

```
__property TFileName ChangedTableName;
```

Description

Set ChangedTableName to create a table that saves all unchanged versions of records that are changed by the batch operation. ChangedTableName must specify the name of a Interbase table. Setting ChangedTableName ensures that no information in the Destination (see page 32) table gets lost when the records are altered or deleted.

When the Mode (see page 34) property is batUpdate or batAppendUpdate, the new table will hold all records that were updated. When the Mode (see page 34) property is batDelete, the new table will hold all records that were deleted.

The ChangedCount (see page 31) property specifies the number of records in the new table.

2.1.1.5.5 TIBBatchMove.CommitCount Property

Specifies how many records are batch moved before a commit occurs.

Pascal

```
property CommitCount: Integer;
```

C++

```
__property int CommitCount;
```

Description

Use CommitCount to chunk batch operations into smaller operations. CommitCount allows an application to accommodate server transaction logs that are not big enough to handle the entire batch. Setting CommitCount to 0 will cause chunking into the number of records that will fit in 32K bytes.

Note: If the Execute (see page 28) method is called within an explicit transaction on the database containing the tables, CommitCount is ignored.

2.1.1.5.6 TIBBatchMove.Destination Property

Specifies the TIBTable (see page 600) object for the database table that is the destination of the batch operation.

Pascal

```
property Destination: TIBTable;
```

C++

```
__property TIBTable Destination;
```

Description

Use Destination to indicate the database table that will be created or altered by the batch operation. The batch operation affects the Destination differently depending on the Mode (see page 34):

Mode (see page 34) Effect on Destination

- batAppend Records are added to the destination table. No pre-existing records are altered. The destination table must already exist.
- batUpdate Records in the destination table with key fields matching the corresponding fields in the source are changed to match the source records. No new records are added. The destination table must already exist and must have an index defined.
- batAppendUpdate Records in the destination table with key fields matching the corresponding fields in the source are changed to match the source. Source (see page 36) records that do not match key fields are added to the destination table. The destination table must already exist and must have an index defined.
- batDelete Records in the destination table that match the records in the source are deleted. The destination table must already exist and must have an index defined.
- batCopy The destination table is created to match the records in the source. If Destination already exists, it is replaced by the copy of the source.

2.1.1.5.7 TIBBatchMove.KeyViolCount Property

Reports the number of records that could not be replaced, added, or deleted from Destination (see page 32) because of integrity or key violations.

Pascal

```
property KeyViolCount: Longint;
```

C++

```
__property Longint KeyViolCount;
```

Description

Use KeyViolCount to learn the number of records form the Source (see page 36) table that could not be applied to the Destination (see page 32) table because of integrity or key violations. To obtain a table of the offending records from the Source (see page 36) table, specify a KeyViolTableName (see page 33).

Key violations occur when appending records to a table that has a primary key. On a table with a primary key, no two records may have the same values on all of the key fields. If a record that is being appended has the same values on all the key fields as a record in the destination table, it cannot be appended.

Integrity violations occur when deleting, appending or updating records to a table that has a referential integrity specification. A referential integrity specification requires that certain fields in one table must have values that exist in a particular field of another table. If a record that is being added does not match the restrictions imposed by referential integrity, it cannot be added. If deleting record would cause tighter referential integrity restrictions that another table cannot meet, it cannot be deleted.

Note: If AbortOnKeyViol (see page 30) is true, KeyViolCount will never be greater than one, because the first violation will cause the batch operation to terminate.

2.1.1.5.8 TIBBatchMove.KeyViolTableName Property

Specifies the name of an InterBase table that will be created to contain all records from the Source (see page 36) that cannot participate in the batch operation because of integrity or key violations.

Pascal

```
property KeyViolTableName: TFileName;
```

C++

```
__property TFileName KeyViolTableName;
```

Description

Use KeyViolTableName to obtain a table of records that could not be applied to the Destination (see page 32) table because of integrity or key violations. KeyViolCount (see page 32) has the number of records in the table that is created.

Key violations occur when appending records to a table that has a primary key. On a table with a primary key, no two records may have the same values on all of the key fields. If a record that is being appended has the same values on all the key fields as a record in the destination table, it cannot be appended.

Integrity violations occur when deleting, appending or updating records to a table that has a referential integrity specification. A referential integrity specification requires that certain fields in one table must have values that exist in a particular field of another table. If a record that is being added does not match the restrictions imposed by referential integrity, it cannot be added. If deleting record would cause tighter referential integrity restrictions that another table cannot meet, it cannot be deleted.

Note: If AbortOnKeyViol (see page 30) is true, then there will be at most one record in this table because the operation aborts with that first record.

2.1.1.5.9 TIBBatchMove.Mappings Property

Specifies the column mappings for a batch operation.

Pascal

```
property Mappings: TStrings;
```

C++

```
__property TStrings Mappings;
```

Description

Set Mappings to specify the correspondence between fields in the Source (see page 36) and fields in the Destination (see page 32). By default TIBBatchMove (see page 26) matches fields based on their position in the source and destination tables. That is, the first column in the source is matched with the first column in the destination, and so on. Mappings enables an application to override this default.

Mappings is a list of column mappings (one per line) in one of two forms. To map the column ColName in the source table to the column of the same name in the destination table, use:

```
ColName
```

To map the column named SourceColName in the source table to the column named DestColName in the destination table, use:

```
DestColName=SourceColName
```

When adding or appending records, fields in Destination (see page 32) which have no entry in Mappings will be set to NULL. When copying a dataset, fields in Destination (see page 32) which have no entry in Mappings will not appear as columns in the copy of the table.

If source and destination column data types are not the same, TIBBatchMove (see page 26) can either cancel the batch operation or perform a 'best fit.' If AbortOnProblem (see page 30) is true, type mismatches will terminate the batch move. Otherwise, the field values from the Source (see page 36) will be converted, if possible, to be compatible with the Destination (see page 32).

2.1.1.5.10 TIBBatchMove.Mode Property

Specifies what the TIBBatchMove (see page 26) object does when the Execute (see page 28) method is called.

Pascal

```
property Mode: TBatchMode;
```

C++

```
__property TBatchMode Mode;
```

Description

Use Mode to indicate whether the TIBBatchMove (see page 26) object should add records, replace records, delete records, or copy the Source (see page 36). These are the possible values for Mode:

Value	Meaning
batAppend	Append the records in the source to the destination table. The destination table must already exist, and the two tables must not have records with duplicate keys. This is the default mode.
batUpdate	Replace records in the destination table with matching records from the source table. The destination table must exist and must have an index defined to match records.
batAppendUpdate	If a matching record exists in the destination table, replace it. Otherwise, append records to the destination table. The destination table must exist and must have an index defined to match records.
batCopy	Create (see page 27) the destination table based on the structure of the source table. If the destination already exists, the operation will delete it, and replace it with the new copy of the source.
batDelete	Delete records in the destination table that match records in the source table. The destination table must already exist and must have an index defined.

2.1.1.5.11 TIBBatchMove.MovedCount Property

Reports the number of records form the Source (see page 36) which were applied to the destination.

Pascal

```
property MovedCount: Longint;
```

C++

```
__property Longint MovedCount;
```

Description

Read MovedCount to learn the number of records from the Source (see page 36) that were read during the Execute (see page 28) method. The MovedCount value includes any records which caused key or integrity violations, or which were trimmed.

2.1.1.5.12 TIBBatchMove.ProblemCount Property

Indicates the number of records which could not be added to Destination (see page 32) without loss of data due to a field type mismatch.

Pascal

```
property ProblemCount: Longint;
```

C++

```
__property Longint ProblemCount;
```

Description

Read ProblemCount to learn the number of records from the Source (see page 36) that had field values that could not be mapped to destination fields without being 'trimmed'. If a ProblemTableName (see page 35) is specified, ProblemCount is the number of records in that table.

Trimming a field means converting its value to a value compatible with the destination field type. For example, if a field in the destination table holds a string of size 10 characters, and the corresponding field in the source table holds a string of size 15 characters, any values from the source table longer than 10 characters must be truncated. Values that cannot be converted will cause TIBBatchMove (see page 26) to raise an exception.

Trimming does not occur when the Mode (see page 34) property is set to batCopy. Trimming fields on a delete operation means that records in the destination table may be deleted that do not exactly match the values from the source table.

Note: If AbortOnProblem (see page 30) is true, then ProblemCount will be at most one, because the operation aborts when the problem occurs.

2.1.1.5.13 TIBBatchMove.ProblemTableName Property

Specifies the name of a Interbase table that will be created to contain all records from the Source (see page 36) that contain fields which were trimmed to match the corresponding field type in the Destination (see page 32) table.

Pascal

```
property ProblemTableName: TFileName;
```

C++

```
__property TFileName ProblemTableName;
```

Description

Set ProblemTableName to create a table that holds records from the Source (see page 36) that contain fields that had to be trimmed to be applied to the Destination (see page 32). ProblemCount (see page 35) will have the number of records placed in the new table. If ProblemTableName is not specified, the data in the record is still trimmed when applied to the destination table, but there will be no easy way to tell which fields were trimmed.

Trimming a field means converting its value to a value compatible with the destination field type. For example, if a field in the destination table holds a string of size 10 characters, and the corresponding field in the source table holds a string of size 15 characters, any values from the source table longer than 10 characters must be truncated. Values that cannot be converted will cause TIBBatchMove (see page 26) to raise an exception.

Trimming does not occur when the Mode (see page 34) property is set to batCopy. Trimming fields on a delete operation means that records in the destination table may be deleted that do not exactly match the values from the source table.

Note: If AbortOnProblem (see page 30) is true, then there will be at most one record in this table because the operation aborts with the first record that has a problem.

2.1.1.5.14 TIBBatchMove.RecordCount Property

Specifies the maximum number of records that are applied to the Destination (see page 32) when Execute (see page 28) is called.

Pascal

```
property RecordCount: Longint;
```

C++

```
__property Longint RecordCount;
```

Description

Use RecordCount to limit the number of Source (see page 36) records that are applied to the Destination (see page 32). If zero, all records are added, subtracted, or copied, beginning with the first record in Source (see page 36). If RecordCount is greater than zero, up to RecordCount records are applied to the Destination (see page 32), starting with the current record in the dataset. If RecordCount exceeds the number of records remaining in Source (see page 36), no wraparound occurs; the operation is terminated.

2.1.1.5.15 TIBBatchMove.Source Property

Specifies the dataset that is the source of the batch operation.

Pascal

```
property Source: TIBCustomDataSet;
```

C++

```
__property TIBCustomDataSet Source;
```

Description

Use Source to indicate the dataset that contains the records that will be added to, subtracted from, or copied to the Destination (see page 32). If Source is a TIBQuery component, the query is executed before the move is executed. If a filter is currently in effect on Source, the filter conditions are respected when the batch move executes. The Mode (see page 34) property indicates how the Source records are applied to the Destination (see page 32):

Mode (see page 34)	Effect on Destination (see page 32)
batAppend	batAppend Records are added to the destination table.
batUpdate	Records in the destination table with key fields matching the corresponding fields in the source are changed to match the source records.
batAppendUpdate	Records in the destination table with key fields matching the corresponding fields in the source are changed to match the source. Source records that do not match key fields are added to the destination table.

batDelete	Records in the destination table that match the records in the source are deleted.
batCopy	Destination (see page 32) table is created to match the records in the source.

2.1.1.5.16 TIBBatchMove.Transliterate Property

Specifies whether the data in the Source (see page 36) records should be converted form the locale of the Source (see page 36) to the locale of the Destination (see page 32) when the Execute (see page 28) method is called.

Pascal

```
property Transliterate: Boolean;
```

C++

```
__property Boolean Transliterate;
```

Description

Set Transliterate to true when the Source (see page 36) dataset and the Destination (see page 32) table use different language drivers and the data may contain extended ASCII characters. Set Transliterate to false to avoid the overhead of the character set conversion when both the Source (see page 36) and the Destination (see page 32) use the same language driver or the data does not contain extended ASCII characters.

2.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TBatchMode (see page 38)	This is record IBBatchMove.TBatchMode.

2.2.1 IBBatchMove.TBatchMode Enumeration

Pascal

```
TBatchMode = (
  batAppend,
  batUpdate,
  batAppendUpdate,
  batDelete,
  batCopy
);
```

C++

```
enum TBatchMode {
  batAppend,
  batUpdate,
  batAppendUpdate,
  batDelete,
  batCopy
};
```

File

IBBatchMove

Description

This is record IBBatchMove.TBatchMode.

3 IBBlob Namespace

This is namespace IBBlob.

Topics

Name	Description
Classes (see page 40)	The following table lists classes in this documentation.
Functions (see page 54)	The following table lists functions in this documentation.
Constants (see page 56)	The following table lists constants in this documentation.

Classes

	Name	Description
	TIBBlobStream (see page 40)	TIBBlobStream is a stream object that lets applications read from or write to field objects that represent Blob fields.

Constants

Name	Description
DefaultBlobSegmentSize (see page 56)	This is constant IBBlob.DefaultBlobSegmentSize.

Functions

	Name	Description
	GetBlobInfo (see page 54)	This is function IBBlob.GetBlobInfo.
	ReadBlob (see page 54)	This is function IBBlob.ReadBlob.
	WriteBlob (see page 55)	This is function IBBlob.WriteBlob.

3.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBBlobStream (see page 40)	TIBBlobStream is a stream object that lets applications read from or write to field objects that represent Blob fields.

3.1.1 TIBBlobStream Class

TIBBlobStream is a stream object that lets applications read from or write to field objects that represent Blob fields.

Class Hierarchy

Pascal

```
TIBBlobStream = class(TStream);
```

C++

```
class TIBBlobStream : public TStream;
```

File

IBBlob

Description

Use TIBBlobStream to access or modify the value of a Blob field object. Blob field objects are TBlobField objects and descendants of TBlobField such as TGraphicField and TMemoField. Blob fields use Blob streams to implement many of their data access properties and methods.

TIBBlobStream allows objects that have no specialized knowledge of how data is stored in a Blob field to read or write such data by employing the uniform stream interface.

To use a Blob stream, create an instance of TIBBlobStream, use the methods of the stream to read or write the data, and then free the Blob stream. Do not use the same instance of TBlobStream to access data from more than one record. Instead, create a new TIBBlobStream object every time you need to read or write Blob data on a new record.

3.1.1.1 TIBBlobStream Members

The following tables list the members exposed by TIBBlobStream.

Protected Methods

	Name	Description
	CloseBlob (see page 44)	This is CloseBlob, a member of class TIBBlobStream.
	CreateBlob (see page 44)	This is CreateBlob, a member of class TIBBlobStream.
	EnsureBlobInitialized (see page 44)	This is EnsureBlobInitialized, a member of class TIBBlobStream.
	GetBlobInfo (see page 45)	This is GetBlobInfo, a member of class TIBBlobStream.

	GetDatabase (see page 45)	This is GetDatabase, a member of class TIBBlobStream.
	GetDBHandle (see page 45)	This is GetDBHandle, a member of class TIBBlobStream.
	GetTransaction (see page 45)	This is GetTransaction, a member of class TIBBlobStream.
	GetTRHandle (see page 46)	This is GetTRHandle, a member of class TIBBlobStream.
	OpenBlob (see page 46)	This is OpenBlob, a member of class TIBBlobStream.
	SetBlobID (see page 48)	This is SetBlobID, a member of class TIBBlobStream.
	SetDatabase (see page 48)	This is SetDatabase, a member of class TIBBlobStream.
	SetMode (see page 49)	This is SetMode, a member of class TIBBlobStream.
	SetTransaction (see page 49)	This is SetTransaction, a member of class TIBBlobStream.

Public Methods

	Name	Description
	Call (see page 43)	Returns an error message based on the error code.
	Cancel (see page 43)	Cancels all changes to the BLOB field, reverting to its state when the Blob stream was created.
	CheckReadable (see page 43)	Indicates whether or not a Blob is readable.
	CheckWritable (see page 44)	Indicates whether or not a Blob is write-able.
	Create (see page 41)	Creates an instance of TIBBlobStream (see page 40).
V	Destroy (see page 42)	Destroys an instance of TIBBlobStream (see page 40).
	Finalize (see page 45)	Creates a Blob on the InterBase server and writes the data from the BlobStream to it.
	LoadFromFile (see page 46)	Loads a Blob from a file to the field.
	LoadFromStream (see page 46)	Loads a Blob from a stream into the field.
V	Read (see page 47)	Reads up to Count bytes from the current position in the field's data into Buffer.
	SaveToFile (see page 47)	Saves the contents of the Blob field to a file.
	SaveToStream (see page 47)	Saves the contents of the BLOB field to a stream.
V	Seek (see page 48)	Resets the current position of the TIBBlobStream (see page 40) object.
V	SetSize (see page 49)	Set the new total size of the Blob.
	Truncate (see page 49)	Discards all data in the Blob field from the current position on.
V	Write (see page 50)	Writes Count bytes from Buffer to the current position in the field and updates the current position by Count bytes.

Public Properties

	Name	Description
	BlobID (see page 50)	Sets or returns the Blob ID.
R	BlobMaxSegmentSize (see page 51)	Returns the maximum segment size.
R	BlobNumSegments (see page 51)	Returns the total number of segments in the Blob.
R	BlobSize (see page 51)	Returns the total size of the Blob.
R	BlobType (see page 51)	Returns the Blob type.
	Database (see page 52)	Sets or returns the associated database.
R	DBHandle (see page 52)	Indicates the database handle.
R	Handle (see page 52)	Indicates the Blob handle.
	Mode (see page 52)	Sets or returns the Blob stream mode type.
R	Modified (see page 53)	Indicates whether the Blob field has been modified.
	Transaction (see page 53)	Gets or sets the associated transaction.
R	TRHandle (see page 53)	Indicates the transaction handle.

3.1.1.2 TIBBlobStream.Create Constructor

Creates an instance of TIBBlobStream (see page 40).

Pascal

```
constructor Create;
```

C++

```
__fastcall TIBBlobStream();
```

Description

Call (see page 43) Create to obtain an instance of TIBBlobStream (see page 40) for reading from or writing to a specific TBlobField object.

3.1.1.3 TIBBlobStream.Destroy Destructor

Destroys an instance of TIBBlobStream (see page 40).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBBlobStream();
```

Description

Do not call Destroy directly in an application. Instead call Free. Free verifies that the TIBBlobStream (see page 40) object is not already freed and only then calls Destroy.

Destroy the TIBBlobStream (see page 40) object by calling Free when it is no longer needed for reading from or writing to the BLOB field.

Destroy triggers an OnDataChange event if the Blob stream was used to overwrite or modify the data in the field. Destroy then frees any buffers that were allocated to handle the data.

3.1.1.4 TIBBlobStream Methods

The methods of the TIBBlobStream class are listed here.

Protected Methods

	Name	Description
	CloseBlob (see page 44)	This is CloseBlob, a member of class TIBBlobStream.
	CreateBlob (see page 44)	This is CreateBlob, a member of class TIBBlobStream.
	EnsureBlobInitialized (see page 44)	This is EnsureBlobInitialized, a member of class TIBBlobStream.
	GetBlobInfo (see page 45)	This is GetBlobInfo, a member of class TIBBlobStream.
	GetDatabase (see page 45)	This is GetDatabase, a member of class TIBBlobStream.
	GetDBHandle (see page 45)	This is GetDBHandle, a member of class TIBBlobStream.
	GetTransaction (see page 45)	This is GetTransaction, a member of class TIBBlobStream.
	GetTRHandle (see page 46)	This is GetTRHandle, a member of class TIBBlobStream.
	OpenBlob (see page 46)	This is OpenBlob, a member of class TIBBlobStream.
	SetBlobID (see page 48)	This is SetBlobID, a member of class TIBBlobStream.
	SetDatabase (see page 48)	This is SetDatabase, a member of class TIBBlobStream.
	SetMode (see page 49)	This is SetMode, a member of class TIBBlobStream.
	SetTransaction (see page 49)	This is SetTransaction, a member of class TIBBlobStream.

Public Methods

	Name	Description
	Call (see page 43)	Returns an error message based on the error code.
	Cancel (see page 43)	Cancels all changes to the BLOB field, reverting to its state when the Blob stream was created.

	CheckReadable (see page 43)	Indicates whether or not a Blob is readable.
	CheckWritable (see page 44)	Indicates whether or not a Blob is write-able.
	Finalize (see page 45)	Creates a Blob on the InterBase server and writes the data from the BlobStream to it.
	LoadFromFile (see page 46)	Loads a Blob from a file to the field.
	LoadFromStream (see page 46)	Loads a Blob from a stream into the field.
V	Read (see page 47)	Reads up to Count bytes from the current position in the field's data into Buffer.
	SaveToFile (see page 47)	Saves the contents of the Blob field to a file.
	SaveToStream (see page 47)	Saves the contents of the BLOB field to a stream.
V	Seek (see page 48)	Resets the current position of the TIBBlobStream (see page 40) object.
V	SetSize (see page 49)	Set the new total size of the Blob.
	Truncate (see page 49)	Discards all data in the Blob field from the current position on.
V	Write (see page 50)	Writes Count bytes from Buffer to the current position in the field and updates the current position by Count bytes.

3.1.1.4.1 TIBBlobStream.Call Method

Returns an error message based on the error code.

Pascal

```
function Call(ErrCode: ISC_STATUS; RaiseError: Boolean): ISC_STATUS;
```

C++

```
__fastcall ISC_STATUS Call(ISC_STATUS ErrCode, Boolean RaiseError);
```

Description

Call is an internal method used to make calls to the InterBase API, and gives you the option of raising an exception or returning an error based on the value of RaiseError.

3.1.1.4.2 TIBBlobStream.Cancel Method

Cancels all changes to the BLOB field, reverting to its state when the Blob stream was created.

Pascal

```
procedure Cancel;
```

C++

```
__fastcall Cancel();
```

Description

Call (see page 43) Cancel to back out all changes to the Blob field using this Blob stream. After a call to Cancel, the Modified (see page 53) property is reset to false.

3.1.1.4.3 TIBBlobStream.CheckReadable Method

Indicates whether or not a Blob is readable.

Pascal

```
procedure CheckReadable;
```

C++

```
__fastcall CheckReadable();
```

Description

Call (see page 43) CheckReadable to determine whether or not a Blob is readable. This method raises an exception if the Blob is not readable.

3.1.1.4.4 TIBBlobStream.CheckWritable Method

Indicates whether or not a Blob is write-able.

Pascal

```
procedure CheckWritable;
```

C++

```
__fastcall CheckWritable();
```

Description

Call (see page 43) CheckWritable to determine whether or not a Blob is writeable. This method raises an exception if the Blob is not writeable.

3.1.1.4.5 TIBBlobStream.CloseBlob Method

Pascal

```
procedure CloseBlob;
```

C++

```
__fastcall CloseBlob();
```

Description

This is CloseBlob, a member of class TIBBlobStream.

3.1.1.4.6 TIBBlobStream.CreateBlob Method

Pascal

```
procedure CreateBlob;
```

C++

```
__fastcall CreateBlob();
```

Description

This is CreateBlob, a member of class TIBBlobStream.

3.1.1.4.7 TIBBlobStream.EnsureBlobInitialized Method

Pascal

```
procedure EnsureBlobInitialized;
```

C++

```
__fastcall EnsureBlobInitialized();
```

Description

This is EnsureBlobInitialized, a member of class TIBBlobStream.

3.1.1.4.8 TIBBlobStream.Finalize Method

Creates a Blob on the InterBase server and writes the data from the BlobStream to it.

Pascal

```
procedure Finalize;
```

C++

```
__fastcall Finalize();
```

Description

Finalize creates a Blob on the InterBase server and writes the data from the BlobStream to the Blob and then closes the Blob.

3.1.1.4.9 TIBBlobStream.GetBlobInfo Method

Pascal

```
procedure GetBlobInfo;
```

C++

```
__fastcall GetBlobInfo();
```

Description

This is GetBlobInfo, a member of class TIBBlobStream.

3.1.1.4.10 TIBBlobStream.GetDatabase Method

Pascal

```
function GetDatabase: TIBDatabase;
```

C++

```
__fastcall TIBDatabase GetDatabase();
```

Description

This is GetDatabase, a member of class TIBBlobStream.

3.1.1.4.11 TIBBlobStream.GetDBHandle Method

Pascal

```
function GetDBHandle: PISC_DB_HANDLE;
```

C++

```
__fastcall PISC_DB_HANDLE GetDBHandle();
```

Description

This is GetDBHandle, a member of class TIBBlobStream.

3.1.1.4.12 TIBBlobStream.GetTransaction Method

Pascal

```
function GetTransaction: TIBTransaction;
```

C++

```
__fastcall TIBTransaction GetTransaction();
```

Description

This is GetTransaction, a member of class TIBBlobStream.

3.1.1.4.13 TIBBlobStream.GetTRHandle Method

Pascal

```
function GetTRHandle: PISC_TR_HANDLE;
```

C++

```
__fastcall PISC_TR_HANDLE GetTRHandle();
```

Description

This is GetTRHandle, a member of class TIBBlobStream.

3.1.1.4.14 TIBBlobStream.LoadFromFile Method

Loads a Blob from a file to the field.

Pascal

```
procedure LoadFromFile(Filename: string);
```

C++

```
__fastcall LoadFromFile(UnicodeString Filename);
```

Description

Use LoadFromFile to load the contents of a file into a Blob field. Specify the name of the file to load into the field as the value of the FileName parameter.

3.1.1.4.15 TIBBlobStream.LoadFromStream Method

Loads a Blob from a stream into the field.

Pascal

```
procedure LoadFromStream(Stream: TStream);
```

C++

```
__fastcall LoadFromStream(TStream Stream);
```

Description

Use LoadFromStream to copy the contents of a stream into the Blob field. Specify the stream from which the field's value is copied as the value of the Stream parameter.

3.1.1.4.16 TIBBlobStream.OpenBlob Method

Pascal

```
procedure OpenBlob;
```

C++

```
__fastcall OpenBlob();
```

Description

This is OpenBlob, a member of class TIBBlobStream.

3.1.1.4.17 TIBBlobStream.Read Method

Reads up to Count bytes from the current position in the field's data into Buffer.

Pascal

```
function Read(var Buffer; Count: Longint): Longint; override;
```

C++

```
virtual __fastcall Longint Read( Buffer, Longint Count);
```

Description

Call (see page 43) Read to read data from the Blob field when the number of bytes in the field's data is not known. Buffer must have at least Count bytes allocated to hold the data that was read from the field.

Read transfers up to Count bytes from the Blob data into Buffer, starting in the current position, and then advances the current position by the number of bytes actually transferred. Read returns the number of bytes actually transferred (which may be less than the number requested in Count.)

Read checks the Transliterate property of the field, and converts the data into ANSI from the character set specified by the dataset if Transliterate is true.

All the other data-reading methods of a Blob stream (ReadBuffer, ReadComponent) call Read to do their actual reading.

Do not call Read when the TIBBlobStream (see page 40) was created in bmWrite mode.

3.1.1.4.18 TIBBlobStream.SaveToFile Method

Saves the contents of the Blob field to a file.

Pascal

```
procedure SaveToFile(Filename: string);
```

C++

```
__fastcall SaveToFile(UnicodeString Filename);
```

Description

Use SaveToFile to save the contents of the Blob field to a file. Specify the name of the file as the value of the FileName parameter.

3.1.1.4.19 TIBBlobStream.SaveToStream Method

Saves the contents of the BLOB field to a stream.

Pascal

```
procedure SaveToStream(Stream: TStream);
```

C++

```
__fastcall SaveToStream(TStream Stream);
```

Description

Use SaveToStream to copy the contents of a Blob field to a stream. Specify the name of the stream to which the field's value is

saved as the value of the Stream parameter.

3.1.1.4.20 TIBBlobStream.Seek Method

Resets the current position of the TIBBlobStream (see page 40) object.

Pascal

```
function Seek(Offset: Longint; Origin: Word): Longint; override;
```

C++

```
virtual __fastcall Longint Seek(Longint Offset, Word Origin);
```

Description

Use Seek to move the current position within the Blob data by the indicated offset. Seek allows an application to read from or write to a particular location within the Blob data.

The Origin parameter indicates how to interpret the Offset parameter. Origin should be one of the following values:

Value	Meaning
soFromBeginning	Offset is from the beginning of the Blob data Seek moves to the position Offset Offset must be >= 0
soFromCurrent	Offset is from the current position in the Blob data Seek moves to Position + Offset
soFromEnd	Offset is from the end of the Blob data Offset must be <= 0 to indicate a number of bytes before the end of the Blob

Seek returns the new value of the Position property, the new current position in the Blob data.

3.1.1.4.21 TIBBlobStream.SetBlobID Method

Pascal

```
procedure SetBlobID(Value: TISC_QUAD);
```

C++

```
__fastcall SetBlobID(TISC_QUAD Value);
```

Description

This is SetBlobID, a member of class TIBBlobStream.

3.1.1.4.22 TIBBlobStream.SetDatabase Method

Pascal

```
procedure SetDatabase(Value: TIBDatabase);
```

C++

```
__fastcall SetDatabase(TIBDatabase Value);
```

Description

This is SetDatabase, a member of class TIBBlobStream.

3.1.1.4.23 TIBBlobStream.SetMode Method

Pascal

```
procedure SetMode(Value: TBlobStreamMode);
```

C++

```
__fastcall SetMode(TBlobStreamMode Value);
```

Description

This is SetMode, a member of class TIBBlobStream.

3.1.1.4.24 TIBBlobStream.SetSize Method

Set the new total size of the Blob.

Pascal

```
procedure SetSize(NewSize: Long); override;
```

C++

```
virtual __fastcall SetSize(Long NewSize);
```

Description

Call (see page 43) SetSize to set the new total size of the Blob.

3.1.1.4.25 TIBBlobStream.SetTransaction Method

Pascal

```
procedure SetTransaction(Value: TIBTransaction);
```

C++

```
__fastcall SetTransaction(TIBTransaction Value);
```

Description

This is SetTransaction, a member of class TIBBlobStream.

3.1.1.4.26 TIBBlobStream.Truncate Method

Discards all data in the Blob field from the current position on.

Pascal

```
procedure Truncate;
```

C++

```
__fastcall Truncate();
```

Description

Use Truncate to limit the size of the Blob data. Calling Truncate when the current position is 0 will clear the contents of the Blob field.

Do not call Truncate when the TIBBlobStream (see page 40) was created in bmRead mode.

3.1.1.4.27 TIBBlobStream.Write Method

Writes Count bytes from Buffer to the current position in the field and updates the current position by Count bytes.

Pascal

```
function Write(const Buffer; Count: Longint): Longint; override;
```

C++

```
virtual __fastcall Longint Write(const  Buffer, Longint Count);
```

Description

Use Write to write Count bytes to the Blob field, starting at the current position. The Write method for TIBBlobStream (see page 40) always writes the entire Count bytes, as Blob data does not necessarily include a termination character. Thus, Write is equivalent to the WriteBuffer method.

Write checks the Transliterate property of the field, and converts the data from ANSI into the character set of the dataset if Transliterate is true.

All the other data-writing methods of a Blob stream (WriteBuffer, WriteComponent) call Write to do their actual writing.

Do not call Write when the TIBBlobStream (see page 40) was created in bmRead mode.

3.1.1.5 TIBBlobStream Properties

The properties of the TIBBlobStream class are listed here.

Public Properties

	Name	Description
	BlobID (see page 50)	Sets or returns the Blob ID.
R	BlobMaxSegmentSize (see page 51)	Returns the maximum segment size.
R	BlobNumSegments (see page 51)	Returns the total number of segments in the Blob.
R	BlobSize (see page 51)	Returns the total size of the Blob.
R	BlobType (see page 51)	Returns the Blob type.
	Database (see page 52)	Sets or returns the associated database.
R	DBHandle (see page 52)	Indicates the database handle.
R	Handle (see page 52)	Indicates the Blob handle.
	Mode (see page 52)	Sets or returns the Blob stream mode type.
R	Modified (see page 53)	Indicates whether the Blob field has been modified.
	Transaction (see page 53)	Gets or sets the associated transaction.
R	TRHandle (see page 53)	Indicates the transaction handle.

3.1.1.5.1 TIBBlobStream.BlobID Property

Sets or returns the Blob ID.

Pascal

```
property BlobID: TISC_QUAD;
```

C++

```
__property TISC_QUAD BlobID;
```

Description

Use BlobID to set or return the 64-bit system-defined Blob ID, which is stored in a field in the table and points to the first segment of the Blob or to a page of pointers to Blob fragments

3.1.1.5.2 TIBBlobStream.BlobMaxSegmentSize Property

Returns the maximum segment size.

Pascal

```
property BlobMaxSegmentSize: Long;
```

C++

```
__property Long BlobMaxSegmentSize;
```

Description

Use BlobMaxSegmentSize to return the length of the longest Blob segment.

3.1.1.5.3 TIBBlobStream.BlobNumSegments Property

Returns the total number of segments in the Blob.

Pascal

```
property BlobNumSegments: Long;
```

C++

```
__property Long BlobNumSegments;
```

Description

Use BlobNumSegments to return the total number of segments in the Blob.

3.1.1.5.4 TIBBlobStream.BlobSize Property

Returns the total size of the Blob.

Pascal

```
property BlobSize: Long;
```

C++

```
__property Long BlobSize;
```

Description

Use BlobSize to return the total size of the Blob in bytes.

3.1.1.5.5 TIBBlobStream.BlobType Property

Returns the Blob type.

Pascal

```
property BlobType: Short;
```

C++

```
__property Short BlobType;
```

Description

Use BlobType to return the Blob type; either 0 for segmented, or 1 for stream.

3.1.1.5.6 TIBBlobStream.Database Property

Sets or returns the associated database.

Pascal

```
property Database: TIBDatabase;
```

C++

```
__property TIBDatabase Database;
```

Description

Use Database to set or return the associated database.

3.1.1.5.7 TIBBlobStream.DBHandle Property

Indicates the database handle.

Pascal

```
property DBHandle: PISC_DB_HANDLE;
```

C++

```
__property PISC_DB_HANDLE DBHandle;
```

Description

Use DBHandle to return the database handle.

3.1.1.5.8 TIBBlobStream.Handle Property

Indicates the Blob handle.

Pascal

```
property Handle: TISC_BLOB_HANDLE;
```

C++

```
__property TISC_BLOB_HANDLE Handle;
```

Description

Use Handle to return the Blob handle.

3.1.1.5.9 TIBBlobStream.Mode Property

Sets or returns the Blob stream mode type.

Pascal

```
property Mode: TBlobStreamMode;
```

C++

```
__property TBlobStreamMode Mode;
```

Description

Use Mode to set or read the Blob stream mode type. The Mode specifies whether the stream is used for reading BLOB values, writing BLOB values, or both.

3.1.1.5.10 TIBBlobStream.Modified Property

Indicates whether the Blob field has been modified.

Pascal

```
property Modified: Boolean;
```

C++

```
__property Boolean Modified;
```

Description

Modified is true when the value to a Blob field has changed. If the value of the Blob field is set using the properties of TBlobField, or using a TIBBlobStream (see page 40) object, Modified is automatically set to true.

3.1.1.5.11 TIBBlobStream.Transaction Property

Gets or sets the associated transaction.

Pascal

```
property Transaction: TIBTransaction;
```

C++

```
__property TIBTransaction Transaction;
```

Description

Use Transaction to set or read the associated transaction.

3.1.1.5.12 TIBBlobStream.TRHandle Property

Indicates the transaction handle.

Pascal

```
property TRHandle: PISC_TR_HANDLE;
```

C++

```
__property PISC_TR_HANDLE TRHandle;
```

Description

Use TRHandle to obtain the transaction handle.

3.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	GetBlobInfo (see page 54)	This is function IBBlob.GetBlobInfo.
	ReadBlob (see page 54)	This is function IBBlob.ReadBlob.
	WriteBlob (see page 55)	This is function IBBlob.WriteBlob.

3.2.1 IBBlob.GetBlobInfo Function

Pascal

```
procedure GetBlobInfo(hBlobHandle: PISC_BLOB_HANDLE; var NumSegments: Long; var
MaxSegmentSize: Long; var TotalSize: Long; var BlobType: Short);
```

C++

```
GetBlobInfo(PISC_BLOB_HANDLE hBlobHandle, Long NumSegments, Long MaxSegmentSize, Long
TotalSize, Short BlobType);
```

File

IBBlob

Description

This is function IBBlob.GetBlobInfo.

3.2.2 IBBlob.ReadBlob Function

Pascal

```
procedure ReadBlob(hBlobHandle: PISC_BLOB_HANDLE; Buffer: PChar; BlobSize: Long);
```

C++

```
ReadBlob(PISC_BLOB_HANDLE hBlobHandle, const char * Buffer, Long BlobSize);
```

File

IBBlob

Description

This is function IBBlob.ReadBlob.

3.2.3 IBBlob.WriteBlob Function

Pascal

```
procedure WriteBlob(hBlobHandle: PISC_BLOB_HANDLE; Buffer: PChar; BlobSize: Long);
```

C++

```
WriteBlob(PISC_BLOB_HANDLE hBlobHandle, const char * Buffer, Long BlobSize);
```

File

IBBlob

Description

This is function IBBlob.WriteBlob.

3.3 Constants

The following table lists constants in this documentation.

Constants

Name	Description
DefaultBlobSegmentSize (see page 56)	This is constant IBBlob.DefaultBlobSegmentSize.

3.3.1 IBBlob.DefaultBlobSegmentSize Constant

Pascal

```
DefaultBlobSegmentSize = 16 * 1024;
```

C++

```
DefaultBlobSegmentSize = 16 * 1024;
```

File

IBBlob

Description

This is constant IBBlob.DefaultBlobSegmentSize.

4 IBCustomDataSet Namespace

This is namespace IBCustomDataSet.

Topics

Name	Description
Classes (see page 59)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 149)	The following table lists structs, records, enums in this documentation.
Types (see page 154)	The following table lists types in this documentation.
Variables (see page 159)	The following table lists variables in this documentation.
Constants (see page 160)	The following table lists constants in this documentation.

Classes

	Name	Description
	TIBBCDField (see page 59)	TIBBCDField represents a numeric field as a Currency type..
	TIBCustomDataSet (see page 62)	TIBCustomDataSet is the base class for all datasets that represent data fetched using InterBase Express.
	TIBDataLink (see page 111)	TIBDataLink is a helper class used by data-aware objects to coordinate the actions of TIBDataSet (see page 114) and to respond to data events.
	TIBDataSet (see page 114)	TIBDataSet executes InterBase SQL statements.
	TIBDataSetUpdateObject (see page 134)	TIBDataSetUpdateObject is the abstract base class for update objects used to update otherwise ununpdateable queries when cached updates are enabled.
	TIBDSBlobStream (see page 138)	TIBDSBlobStream reads from and writes to BLOB fields on behalf of an IB (see page 1) Dataset.
	TIBGeneratorField (see page 142)	TIBGeneratorField calls the server to generate values for a field in an IB (see page 1) dataset.
	TIBStringField (see page 146)	TIBStringField allows for strings in excess of 8196 bytes of data.

Constants

Name	Description
BufferCacheSize (see page 160)	This is constant IBCustomDataSet.BufferCacheSize.
DefaultFieldClasses (see page 160)	This is constant IBCustomDataSet.DefaultFieldClasses.
UniCache (see page 161)	This is constant IBCustomDataSet.UniCache.

Structs, Records, Enums

	Name	Description
	TCachedUpdateStatus (see page 149)	TCachedUpdateStatus indicates the status of a record in a dataset that has cached updates.
	TFieldData (see page 150)	This is record IBCustomDataSet.TFieldData.

	TIBDBKey (see page 150)	TIBDBKey stores the key for a row in an InterBase table.
	TIBGeneratorApplyEvent (see page 151)	TIBGeneratorApplyEvent indicates when a generator should create field values.
	TIBUpdateAction (see page 151)	TIBUpdateAction indicates the action that a dataset should take after an event handler has responded to an update error.
	TLiveMode (see page 152)	This is record IBCustomDataSet.TLiveMode.
	TRecordData (see page 153)	This is record IBCustomDataSet.TRecordData.

Types

Name	Description
PBlobDataArray (see page 154)	This is type IBCustomDataSet.PBlobDataArray.
PDateTime (see page 154)	This is type IBCustomDataSet.PDateTime.
PFieldData (see page 155)	This is type IBCustomDataSet.PFieldData.
PIBDBKey (see page 155)	This is type IBCustomDataSet.PIBDBKey.
PRecordData (see page 155)	This is type IBCustomDataSet.PRecordData.
TBlobDataArray (see page 156)	This is type IBCustomDataSet.TBlobDataArray.
TIBUpdateErrorEvent (see page 156)	TIBUpdateErrorEvent is the type of event handlers that respond when an IBX dataset encounters an error applying a cached update.
TIBUpdateRecordEvent (see page 157)	TIBUpdateRecordEvent is the type of event handlers that respond when an IBX dataset applies a cached update.
TIBUpdateRecordTypes (see page 157)	TIBUpdateRecordTypes is a subset of all possible record types in a dataset that caches updates.
TLiveModes (see page 157)	TLiveModes and TLiveModes indicate what operations an application can perform on an IBX dataset.

Variables

Name	Description
CreateProviderProc (see page 159)	This is variable IBCustomDataSet.CreateProviderProc.

4.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBBCDField (see page 59)	TIBBCDField represents a numeric field as a Currency type..
	TIBCustomDataSet (see page 62)	TIBCustomDataSet is the base class for all datasets that represent data fetched using InterBase Express.
	TIBDataLink (see page 111)	TIBDataLink is a helper class used by data-aware objects to coordinate the actions of TIBDataSet (see page 114) and to respond to data events.
	TIBDataSet (see page 114)	TIBDataSet executes InterBase SQL statements.
	TIBDataSetUpdateObject (see page 134)	TIBDataSetUpdateObject is the abstract base class for update objects used to update otherwise unupdateable queries when cached updates are enabled.
	TIBDSBlobStream (see page 138)	TIBDSBlobStream reads from and writes to BLOB fields on behalf of an IB (see page 1) Dataset.
	TIBGeneratorField (see page 142)	TIBGeneratorField calls the server to generate values for a field in an IB (see page 1) dataset.
	TIBStringField (see page 146)	TIBStringField allows for strings in excess of 8196 bytes of data.

4.1.1 TIBBCDField Class

TIBBCDField represents a numeric field as a Currency type..

Class Hierarchy

Pascal

```
TIBBCDField = class(TBCDField);
```

C++

```
class TIBBCDField : public TBCDField;
```

File

IBCustomDataSet

Description

TIBBCDField encapsulates the Currency type. Despite the name, TIBBCDField does not represent a BCD value. Rather, TIBBCDFields represent InterBase Numeric(x, y) fields, where x < 18 and y <= 4.

4.1.1.1 TIBBCDField Members

The following tables list the members exposed by TIBBCDField.

Protected Methods

	Name	Description
	CheckTypeSize (see page 60)	This is CheckTypeSize, a member of class TIBBCDField.
	GetAsCurrency (see page 61)	This is GetAsCurrency, a member of class TIBBCDField.

	GetAsString (see page 61)	This is GetAsString, a member of class TIBBCDField.
	GetAsVariant (see page 61)	This is GetAsVariant, a member of class TIBBCDField.
	GetDataSize (see page 61)	This is GetDataSize, a member of class TIBBCDField.

Public Methods

	Name	Description
	Create (see page 60)	Creates an instance of a TIBBCDField (see page 59) object.

Published Properties

	Name	Description
	Size (see page 62)	This is Size, a member of class TIBBCDField.

4.1.1.2 TIBBCDField.Create Constructor

Creates an instance of a TIBBCDField (see page 59) object.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBBCDField(TComponent * AOwner);
```

Description

Create sets the datatype to ftBCD and the size to 8.

4.1.1.3 TIBBCDField Methods

The methods of the TIBBCDField class are listed here.

Protected Methods

	Name	Description
	CheckTypeSize (see page 60)	This is CheckTypeSize, a member of class TIBBCDField.
	GetAsCurrency (see page 61)	This is GetAsCurrency, a member of class TIBBCDField.
	GetAsString (see page 61)	This is GetAsString, a member of class TIBBCDField.
	GetAsVariant (see page 61)	This is GetAsVariant, a member of class TIBBCDField.
	GetDataSize (see page 61)	This is GetDataSize, a member of class TIBBCDField.

4.1.1.3.1 TIBBCDField.CheckTypeSize Method

Pascal

```
class procedure CheckTypeSize(Value: Integer); override;
```

C++

```
virtual __fastcall CheckTypeSize(int Value);
```

Description

This is CheckTypeSize, a member of class TIBBCDField.

4.1.1.3.2 TIBBCDField.GetAsCurrency Method

Pascal

```
function GetAsCurrency: Currency; override;
```

C++

```
virtual __fastcall Currency GetAsCurrency();
```

Description

This is GetAsCurrency, a member of class TIBBCDField.

4.1.1.3.3 TIBBCDField.GetAsString Method

Pascal

```
function GetAsString: string; override;
```

C++

```
virtual __fastcall UnicodeString GetAsString();
```

Description

This is GetAsString, a member of class TIBBCDField.

4.1.1.3.4 TIBBCDField.GetAsVariant Method

Pascal

```
function GetAsVariant: Variant; override;
```

C++

```
virtual __fastcall Variant GetAsVariant();
```

Description

This is GetAsVariant, a member of class TIBBCDField.

4.1.1.3.5 TIBBCDField.GetDataSize Method

Pascal

```
function GetDataSize: Integer; override;
```

C++

```
virtual __fastcall int GetDataSize();
```

Description

This is GetDataSize, a member of class TIBBCDField.

4.1.1.4 TIBBCDField Properties

The properties of the TIBBCDField class are listed here.

Published Properties

	Name	Description
	Size (see page 62)	This is Size, a member of class TIBBCDField.

4.1.1.4.1 TIBBCDField.Size Property

Pascal

```
property Size;
```

C++

```
__property Size;
```

Description

This is Size, a member of class TIBBCDField.

4.1.2 TIBCustomDataSet Class

TIBCustomDataSet is the base class for all datasets that represent data fetched using InterBase Express.

Class Hierarchy

Pascal

```
TIBCustomDataSet = class(TDataset);
```

C++

```
class TIBCustomDataSet : public TDataset;
```

File

IBCustomDataSet

Description

TIBCustomDataSet is a dataset object that defines InterBase Express (IBX) functionality for a dataset. Applications never use TIBCustomDataSet objects directly. Instead they use the descendants of TIBCustomDataSet, such as TIBDataSet (see page 114), TIBQuery, TIBStoredProc, and TIBTable, which inherit its dataset-related properties and methods.

See Also

Getting Started with InterBase Express

4.1.2.1 TIBCustomDataSet Members

The following tables list the members exposed by TIBCustomDataSet.

Protected Events

	Name	Description
	AfterDatabaseDisconnect (see page 109)	Occurs immediately after the connection to the associated database is closed.
	AfterTransactionEnd (see page 109)	Occurs immediately after a transaction that includes this dataset's query ends.
	BeforeDatabaseDisconnect (see page 109)	Occurs immediately before the connection to the associated database is closed.
	BeforeTransactionEnd (see page 109)	Occurs immediately before a transaction that includes this dataset's query ends.

	DatabaseFree (see page 110)	Occurs when the database associated with this dataset is freed.
	TransactionFree (see page 111)	Occurs when the transaction that includes this dataset's query is freed.

Protected Methods

	Name	Description
	ActivateConnection (see page 69)	This is ActivateConnection, a member of class TIBCustomDataSet.
	ActivateTransaction (see page 69)	This is ActivateTransaction, a member of class TIBCustomDataSet.
V	AllocRecordBuffer (see page 69)	This is AllocRecordBuffer, a member of class TIBCustomDataSet.
	CheckDatasetClosed (see page 71)	This is CheckDatasetClosed, a member of class TIBCustomDataSet.
	CheckDatasetOpen (see page 71)	This is CheckDatasetOpen, a member of class TIBCustomDataSet.
	CheckNotUniDirectional (see page 71)	This is CheckNotUniDirectional, a member of class TIBCustomDataSet.
V	ClearCalcFields (see page 71)	This is ClearCalcFields, a member of class TIBCustomDataSet.
	ConstraintsStored (see page 72)	This is ConstraintsStored, a member of class TIBCustomDataSet.
V	CreateFields (see page 72)	This is CreateFields, a member of class TIBCustomDataSet.
	DeactivateTransaction (see page 73)	This is DeactivateTransaction, a member of class TIBCustomDataSet.
V	Disconnect (see page 73)	This is Disconnect, a member of class TIBCustomDataSet.
V	DoBeforeDelete (see page 73)	This is DoBeforeDelete, a member of class TIBCustomDataSet.
V	DoBeforeEdit (see page 73)	This is DoBeforeEdit, a member of class TIBCustomDataSet.
V	DoBeforeInsert (see page 74)	This is DoBeforeInsert, a member of class TIBCustomDataSet.
V	DoOnNewRecord (see page 74)	This is DoOnNewRecord, a member of class TIBCustomDataSet.
V	FreeRecordBuffer (see page 75)	This is FreeRecordBuffer, a member of class TIBCustomDataSet.
	GetActiveBuf (see page 75)	This is GetActiveBuf, a member of class TIBCustomDataSet.
V	GetBookmarkData (see page 75)	This is GetBookmarkData, a member of class TIBCustomDataSet.
V	GetBookmarkFlag (see page 75)	This is GetBookmarkFlag, a member of class TIBCustomDataSet.
V	GetCanModify (see page 75)	This is GetCanModify, a member of class TIBCustomDataSet.
V	GetDataSource (see page 76)	This is GetDataSource, a member of class TIBCustomDataSet.
V	GetFieldClass (see page 76)	This is GetFieldClass, a member of class TIBCustomDataSet.
V	GetRecNo (see page 77)	This is GetRecNo, a member of class TIBCustomDataSet.
V	GetRecord (see page 77)	This is GetRecord, a member of class TIBCustomDataSet.
V	GetRecordCount (see page 78)	This is GetRecordCount, a member of class TIBCustomDataSet.
V	GetRecordSize (see page 78)	This is GetRecordSize, a member of class TIBCustomDataSet.
V	InitRecord (see page 78)	This is InitRecord, a member of class TIBCustomDataSet.
V	InternalAddRecord (see page 78)	This is InternalAddRecord, a member of class TIBCustomDataSet.
V	InternalBatchInput (see page 79)	This is InternalBatchInput, a member of class TIBCustomDataSet.
V	InternalBatchOutput (see page 79)	This is InternalBatchOutput, a member of class TIBCustomDataSet.
V	InternalCancel (see page 79)	This is InternalCancel, a member of class TIBCustomDataSet.
V	InternalClose (see page 79)	This is InternalClose, a member of class TIBCustomDataSet.
V	InternalDelete (see page 79)	This is InternalDelete, a member of class TIBCustomDataSet.
V	InternalExecQuery (see page 80)	This is InternalExecQuery, a member of class TIBCustomDataSet.
V	InternalFirst (see page 80)	This is InternalFirst, a member of class TIBCustomDataSet.
V	InternalGotoBookmark (see page 80)	This is InternalGotoBookmark, a member of class TIBCustomDataSet.
V	InternalHandleException (see page 80)	This is InternalHandleException, a member of class TIBCustomDataSet.
V	InternalInitFieldDefs (see page 81)	This is InternalInitFieldDefs, a member of class TIBCustomDataSet.
V	InternalInitRecord (see page 81)	This is InternalInitRecord, a member of class TIBCustomDataSet.
V	InternalInsert (see page 81)	This is InternalInsert, a member of class TIBCustomDataSet.
V	InternalLast (see page 81)	This is InternalLast, a member of class TIBCustomDataSet.
V	InternalOpen (see page 81)	This is InternalOpen, a member of class TIBCustomDataSet.
V	InternalPost (see page 82)	This is InternalPost, a member of class TIBCustomDataSet.
V	InternalPrepare (see page 82)	This is InternalPrepare, a member of class TIBCustomDataSet.
V	InternalRefresh (see page 82)	This is InternalRefresh, a member of class TIBCustomDataSet.
V	InternalRefreshRow (see page 82)	This is InternalRefreshRow, a member of class TIBCustomDataSet.
V	InternalSetParamsFromCursor (see page 83)	This is InternalSetParamsFromCursor, a member of class TIBCustomDataSet.
V	InternalSetToRecord (see page 83)	This is InternalSetToRecord, a member of class TIBCustomDataSet.
V	InternalUnPrepare (see page 83)	This is InternalUnPrepare, a member of class TIBCustomDataSet.

	IsCursorOpen (see page 83)	This is IsCursorOpen, a member of class TIBCustomDataSet.
	Loaded (see page 84)	This is Loaded, a member of class TIBCustomDataSet.
	PSEndTransaction (see page 86)	This is PSEndTransaction, a member of class TIBCustomDataSet.
	PSExecuteStatement (see page 86)	This is PSExecuteStatement, a member of class TIBCustomDataSet.
	PSGetQuoteChar (see page 87)	This is PSGetQuoteChar, a member of class TIBCustomDataSet.
	PSGetTableName (see page 87)	This is PSGetTableName, a member of class TIBCustomDataSet.
	PSGetUpdateException (see page 87)	This is PSGetUpdateException, a member of class TIBCustomDataSet.
	PSInTransaction (see page 87)	This is PSInTransaction, a member of class TIBCustomDataSet.
	PSIsSQLBased (see page 88)	This is PSIsSQLBased, a member of class TIBCustomDataSet.
	PSIsSQLSupported (see page 88)	This is PSIsSQLSupported, a member of class TIBCustomDataSet.
	PSReset (see page 88)	This is PSReset, a member of class TIBCustomDataSet.
	PSStartTransaction (see page 88)	This is PSStartTransaction, a member of class TIBCustomDataSet.
	PSUpdateRecord (see page 88)	This is PSUpdateRecord, a member of class TIBCustomDataSet.
	ReQuery (see page 89)	This is ReQuery, a member of class TIBCustomDataSet.
	SetActive (see page 89)	This is SetActive, a member of class TIBCustomDataSet.
	SetBookmarkData (see page 90)	This is SetBookmarkData, a member of class TIBCustomDataSet.
	SetBookmarkFlag (see page 90)	This is SetBookmarkFlag, a member of class TIBCustomDataSet.
	SetCachedUpdates (see page 90)	This is SetCachedUpdates, a member of class TIBCustomDataSet.
	SetDataSource (see page 90)	This is SetDataSource, a member of class TIBCustomDataSet.
	SetFieldData (see page 90)	This is the overview for the SetFieldData method overload.
	SetRecNo (see page 91)	This is SetRecNo, a member of class TIBCustomDataSet.

Protected Properties

	Name	Description
	BufferChunks (see page 98)	Specifies the dataset buffer chunk size.
	CachedUpdates (see page 98)	Specifies whether cached updates are enabled for a dataset.
	DeleteSQL (see page 99)	Holds the SQL statement used to delete rows from the dataset.
	GeneratorField (see page 100)	Specifies whether the dataset should refresh its data every time it posts a record.
	InsertSQL (see page 100)	Holds the SQL statement used to insert rows into the dataset.
R	InternalPrepared (see page 100)	Indicates whether the dataset has prepared its queries for execution.
R	LiveMode (see page 101)	Indicates what types of edits, if any, can be made to the data in the dataset.
	ModifySQL (see page 101)	Provides the ability to access the SQL object encapsulating the ModifySQL statement.
	ParamCheck (see page 102)	Specifies whether the parameter list for a query is regenerated if the SQL property changes at runtime.
R	Params (see page 103)	Specifies the parameters to use for any parameterized queries.
R	QDelete (see page 103)	Provides the ability to directly access the SQL object encapsulating the DeleteSQL (see page 99) statement.
R	QInsert (see page 104)	Provides the ability to directly access the SQL object encapsulating the InsertSQL (see page 100) statement.
R	QModify (see page 104)	Provides the ability to directly access the SQL object encapsulating the ModifySQL (see page 101) statement.
R	QRefresh (see page 104)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL (see page 105) statement.
R	QSelect (see page 104)	Provides the ability to directly access the SQL object encapsulating the SelectSQL (see page 105) statement.
	RefreshSQL (see page 105)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.
	SelectSQL (see page 105)	Provides the ability to directly access the SQL object encapsulating the SelectSQL statement.
R	SelectStmtHandle (see page 105)	Gets the handle for the SELECT statement.
R	SQLParams (see page 106)	Specifies the parameters to use for any parameterized queries.
R	StatementType (see page 106)	Returns the statement type of the QSelect (see page 104) query.
	UniDirectional (see page 107)	Determines whether bidirectional cursors are enabled for the dataset.
	UpdateMode (see page 107)	Indicates how records are located when updating records.

Public Methods

	Name	Description
	ApplyUpdates (see page 69)	Writes a dataset's pending cached updates to the database.
V	BookmarkValid (see page 70)	Indicates the validity of a specified bookmark.
	CachedUpdateStatus (see page 70)	Returns the status of the cached updates.
	CancelUpdates (see page 70)	Clears all pending cached updates from the cache.
V	CompareBookmarks (see page 71)	Compares two bookmarks.
V	Create (see page 66)	Creates an instance of a TDataSet component.
V	CreateBlobStream (see page 72)	Returns a TIBDSBlobStream (see page 138) object for reading or writing the data in a specified blob field.
	Current (see page 73)	Returns the descriptor area that holds information about the current query.
V	Destroy (see page 66)	Destroys the instance of a dataset component.
	FetchAll (see page 74)	Retrieves all records from the current cursor position to the end of the file and stores them locally.
V	GetCurrentRecord (see page 76)	Retrieves the current record into a buffer.
V	GetFieldData (see page 76)	This is the overview for the GetFieldData method overload.
V	IsSequenced (see page 83)	IsSequenced indicates whether sequence numbers are available for database records.
V	Locate (see page 84)	Searches the dataset for a specified record and makes that record the current record.
	LocateNext (see page 85)	Searches the dataset for the record after a specified record and makes that record the current record.
V	Lookup (see page 85)	Retrieves field values from a record that matches specified search values.
V	Post (see page 86)	Writes a modified record to the database server.
	RecordModified (see page 89)	Sets the record to modified or unmodified.
	RevertRecord (see page 89)	Restores the current record in the dataset to an unmodified state when cached updates are enabled.
	SQLType (see page 91)	Returns the type of query to be executed.
	Undelete (see page 92)	Restores a record deleted from the dataset.
V	UpdateStatus (see page 92)	Reports the update status for the current record.

Public Properties

	Name	Description
R	DBHandle (see page 99)	Specifies the database handle for the dataset.
R	Plan (see page 103)	Indicates the PLAN portion of the dataset's SELECT query.
R	RowsAffected (see page 105)	Indicates the number of rows operated on by the latest query execution.
R	TRHandle (see page 106)	Specifies the transaction handle for the dataset.
	UpdateObject (see page 107)	Specifies the update object component used to update a read-only result set.
	UpdateRecordTypes (see page 108)	Specifies the type of records visible in a dataset when cached updates are enabled.
R	UpdatesPending (see page 108)	Indicates whether the cached updates buffer contains records that are not yet applied.

Published Events

	Name	Description
	OnUpdateError (see page 110)	Occurs if an exception is generated when cached updates are applied to a database.
	OnUpdateRecord (see page 111)	Occurs when cached updates are applied to a record.

Published Properties

	Name	Description
	AfterCancel (see page 94)	This is AfterCancel, a member of class TIBCustomDataSet.
	AfterClose (see page 94)	This is AfterClose, a member of class TIBCustomDataSet.
	AfterDelete (see page 94)	This is AfterDelete, a member of class TIBCustomDataSet.
	AfterEdit (see page 94)	This is AfterEdit, a member of class TIBCustomDataSet.
	AfterInsert (see page 95)	This is AfterInsert, a member of class TIBCustomDataSet.
	AfterOpen (see page 95)	This is AfterOpen, a member of class TIBCustomDataSet.

	AfterPost (see page 95)	This is AfterPost, a member of class TIBCustomDataSet.
	AfterRefresh (see page 95)	This is AfterRefresh, a member of class TIBCustomDataSet.
	AfterScroll (see page 95)	This is AfterScroll, a member of class TIBCustomDataSet.
	AutoCalcFields (see page 96)	This is AutoCalcFields, a member of class TIBCustomDataSet.
	BeforeCancel (see page 96)	This is BeforeCancel, a member of class TIBCustomDataSet.
	BeforeClose (see page 96)	This is BeforeClose, a member of class TIBCustomDataSet.
	BeforeDelete (see page 96)	This is BeforeDelete, a member of class TIBCustomDataSet.
	BeforeEdit (see page 97)	This is BeforeEdit, a member of class TIBCustomDataSet.
	BeforeInsert (see page 97)	This is BeforeInsert, a member of class TIBCustomDataSet.
	BeforeOpen (see page 97)	This is BeforeOpen, a member of class TIBCustomDataSet.
	BeforePost (see page 97)	This is BeforePost, a member of class TIBCustomDataSet.
	BeforeRefresh (see page 97)	This is BeforeRefresh, a member of class TIBCustomDataSet.
	BeforeScroll (see page 98)	This is BeforeScroll, a member of class TIBCustomDataSet.
	Database (see page 99)	Identifies the database component for which this dataset represents one or more tables.
	ForcedRefresh (see page 99)	Specifies whether the dataset should refresh its data every time it posts a record.
	ObjectView (see page 101)	This is ObjectView, a member of class TIBCustomDataSet.
	OnCalcFields (see page 101)	This is OnCalcFields, a member of class TIBCustomDataSet.
	OnDeleteError (see page 102)	This is OnDeleteError, a member of class TIBCustomDataSet.
	OnEditError (see page 102)	This is OnEditError, a member of class TIBCustomDataSet.
	OnNewRecord (see page 102)	This is OnNewRecord, a member of class TIBCustomDataSet.
	OnPostError (see page 102)	This is OnPostError, a member of class TIBCustomDataSet.
	Transaction (see page 106)	Identifies the transaction under which the query executes.

4.1.2.2 TIBCustomDataSet.Create Constructor

Creates an instance of a TDataSet component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBCustomDataSet(TComponent * AOwner);
```

Description

Call Create to instantiate a dataset component at runtime. Ordinarily applications instantiate dataset descendants, such as TIBTable, TIBQuery, TIBDataSet (see page 114), or TIBStoredProc, rather than TIBCustomDataSet (see page 62). These instantiated objects are handled automatically.

On the other hand, applications that create specialized dataset components, such as custom components, may need to instantiate a TIBCustomDataSet (see page 62) component by calling Create. Create:

- Calls the inherited Create for TDataSet.
- Creates the five query objects.
- Creates the base object to hold the reference to the database and transaction.
- Associates database and transaction related events to the base object component.

4.1.2.3 TIBCustomDataSet.Destroy Destructor

Destroys the instance of a dataset component.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBCustomDataSet();
```

Description

Do not call Destroy directly in an application. Usually destruction of datasets is handled automatically by Delphi. If an application creates its own instances of a dataset, however, the application should call Free, which verifies that the dataset component is not already freed before calling Destroy.

Destroy performs the following tasks:

- Frees the associated query objects.
- Frees the base object.
- Frees the associated internal resources.

4.1.2.4 TIBCustomDataSet Methods

The methods of the TIBCustomDataSet class are listed here.

Protected Methods

	Name	Description
	ActivateConnection (see page 69)	This is ActivateConnection, a member of class TIBCustomDataSet.
	ActivateTransaction (see page 69)	This is ActivateTransaction, a member of class TIBCustomDataSet.
V	AllocRecordBuffer (see page 69)	This is AllocRecordBuffer, a member of class TIBCustomDataSet.
	CheckDatasetClosed (see page 71)	This is CheckDatasetClosed, a member of class TIBCustomDataSet.
	CheckDatasetOpen (see page 71)	This is CheckDatasetOpen, a member of class TIBCustomDataSet.
	CheckNotUniDirectional (see page 71)	This is CheckNotUniDirectional, a member of class TIBCustomDataSet.
V	ClearCalcFields (see page 71)	This is ClearCalcFields, a member of class TIBCustomDataSet.
	ConstraintsStored (see page 72)	This is ConstraintsStored, a member of class TIBCustomDataSet.
V	CreateFields (see page 72)	This is CreateFields, a member of class TIBCustomDataSet.
	DeactivateTransaction (see page 73)	This is DeactivateTransaction, a member of class TIBCustomDataSet.
V	Disconnect (see page 73)	This is Disconnect, a member of class TIBCustomDataSet.
V	DoBeforeDelete (see page 73)	This is DoBeforeDelete, a member of class TIBCustomDataSet.
V	DoBeforeEdit (see page 73)	This is DoBeforeEdit, a member of class TIBCustomDataSet.
V	DoBeforeInsert (see page 74)	This is DoBeforeInsert, a member of class TIBCustomDataSet.
V	DoOnNewRecord (see page 74)	This is DoOnNewRecord, a member of class TIBCustomDataSet.
V	FreeRecordBuffer (see page 75)	This is FreeRecordBuffer, a member of class TIBCustomDataSet.
	GetActiveBuf (see page 75)	This is GetActiveBuf, a member of class TIBCustomDataSet.
V	GetBookmarkData (see page 75)	This is GetBookmarkData, a member of class TIBCustomDataSet.
V	GetBookmarkFlag (see page 75)	This is GetBookmarkFlag, a member of class TIBCustomDataSet.
V	GetCanModify (see page 75)	This is GetCanModify, a member of class TIBCustomDataSet.
V	GetDataSource (see page 76)	This is GetDataSource, a member of class TIBCustomDataSet.
V	GetFieldClass (see page 76)	This is GetFieldClass, a member of class TIBCustomDataSet.
V	GetRecNo (see page 77)	This is GetRecNo, a member of class TIBCustomDataSet.
V	GetRecord (see page 77)	This is GetRecord, a member of class TIBCustomDataSet.
V	GetRecordCount (see page 78)	This is GetRecordCount, a member of class TIBCustomDataSet.
V	GetRecordSize (see page 78)	This is GetRecordSize, a member of class TIBCustomDataSet.
V	InitRecord (see page 78)	This is InitRecord, a member of class TIBCustomDataSet.
V	InternalAddRecord (see page 78)	This is InternalAddRecord, a member of class TIBCustomDataSet.
V	InternalBatchInput (see page 79)	This is InternalBatchInput, a member of class TIBCustomDataSet.
V	InternalBatchOutput (see page 79)	This is InternalBatchOutput, a member of class TIBCustomDataSet.

	InternalCancel (see page 79)	This is InternalCancel, a member of class TIBCustomDataSet.
	InternalClose (see page 79)	This is InternalClose, a member of class TIBCustomDataSet.
	InternalDelete (see page 79)	This is InternalDelete, a member of class TIBCustomDataSet.
	InternalExecQuery (see page 80)	This is InternalExecQuery, a member of class TIBCustomDataSet.
	InternalFirst (see page 80)	This is InternalFirst, a member of class TIBCustomDataSet.
	InternalGotoBookmark (see page 80)	This is InternalGotoBookmark, a member of class TIBCustomDataSet.
	InternalHandleException (see page 80)	This is InternalHandleException, a member of class TIBCustomDataSet.
	InternalInitFieldDefs (see page 81)	This is InternalInitFieldDefs, a member of class TIBCustomDataSet.
	InternalInitRecord (see page 81)	This is InternalInitRecord, a member of class TIBCustomDataSet.
	InternalInsert (see page 81)	This is InternalInsert, a member of class TIBCustomDataSet.
	InternalLast (see page 81)	This is InternalLast, a member of class TIBCustomDataSet.
	InternalOpen (see page 81)	This is InternalOpen, a member of class TIBCustomDataSet.
	InternalPost (see page 82)	This is InternalPost, a member of class TIBCustomDataSet.
	InternalPrepare (see page 82)	This is InternalPrepare, a member of class TIBCustomDataSet.
	InternalRefresh (see page 82)	This is InternalRefresh, a member of class TIBCustomDataSet.
	InternalRefreshRow (see page 82)	This is InternalRefreshRow, a member of class TIBCustomDataSet.
	InternalSetParamsFromCursor (see page 83)	This is InternalSetParamsFromCursor, a member of class TIBCustomDataSet.
	InternalSetToRecord (see page 83)	This is InternalSetToRecord, a member of class TIBCustomDataSet.
	InternalUnPrepare (see page 83)	This is InternalUnPrepare, a member of class TIBCustomDataSet.
	IsCursorOpen (see page 83)	This is IsCursorOpen, a member of class TIBCustomDataSet.
	Loaded (see page 84)	This is Loaded, a member of class TIBCustomDataSet.
	PSEndTransaction (see page 86)	This is PSEndTransaction, a member of class TIBCustomDataSet.
	PSExecuteStatement (see page 86)	This is PSExecuteStatement, a member of class TIBCustomDataSet.
	PSGetQuoteChar (see page 87)	This is PSGetQuoteChar, a member of class TIBCustomDataSet.
	PSGetTableName (see page 87)	This is PSGetTableName, a member of class TIBCustomDataSet.
	PSGetUpdateException (see page 87)	This is PSGetUpdateException, a member of class TIBCustomDataSet.
	PSInTransaction (see page 87)	This is PSInTransaction, a member of class TIBCustomDataSet.
	PSIsSQLBased (see page 88)	This is PSIsSQLBased, a member of class TIBCustomDataSet.
	PSIsSQLSupported (see page 88)	This is PSIsSQLSupported, a member of class TIBCustomDataSet.
	PSReset (see page 88)	This is PSReset, a member of class TIBCustomDataSet.
	PSStartTransaction (see page 88)	This is PSStartTransaction, a member of class TIBCustomDataSet.
	PSUpdateRecord (see page 88)	This is PSUpdateRecord, a member of class TIBCustomDataSet.
	ReQuery (see page 89)	This is ReQuery, a member of class TIBCustomDataSet.
	SetActive (see page 89)	This is SetActive, a member of class TIBCustomDataSet.
	SetBookmarkData (see page 90)	This is SetBookmarkData, a member of class TIBCustomDataSet.
	SetBookmarkFlag (see page 90)	This is SetBookmarkFlag, a member of class TIBCustomDataSet.
	SetCachedUpdates (see page 90)	This is SetCachedUpdates, a member of class TIBCustomDataSet.
	SetDataSource (see page 90)	This is SetDataSource, a member of class TIBCustomDataSet.
	SetFieldData (see page 90)	This is the overview for the SetFieldData method overload.
	SetRecNo (see page 91)	This is SetRecNo, a member of class TIBCustomDataSet.

Public Methods

	Name	Description
	ApplyUpdates (see page 69)	Writes a dataset's pending cached updates to the database.
	BookmarkValid (see page 70)	Indicates the validity of a specified bookmark.
	CachedUpdateStatus (see page 70)	Returns the status of the cached updates.
	CancelUpdates (see page 70)	Clears all pending cached updates from the cache.
	CompareBookmarks (see page 71)	Compares two bookmarks.
	CreateBlobStream (see page 72)	Returns a TIBDSBlobStream (see page 138) object for reading or writing the data in a specified blob field.
	Current (see page 73)	Returns the descriptor area that holds information about the current query.
	FetchAll (see page 74)	Retrieves all records from the current cursor position to the end of the file and stores them locally.
	GetCurrentRecord (see page 76)	Retrieves the current record into a buffer.
	GetFieldData (see page 76)	This is the overview for the GetFieldData method overload.

	IsSequenced (see page 83)	IsSequenced indicates whether sequence numbers are available for database records.
	Locate (see page 84)	Searches the dataset for a specified record and makes that record the current record.
	LocateNext (see page 85)	Searches the dataset for the record after a specified record and makes that record the current record.
	Lookup (see page 85)	Retrieves field values from a record that matches specified search values.
	Post (see page 86)	Writes a modified record to the database server.
	RecordModified (see page 89)	Sets the record to modified or unmodified.
	RevertRecord (see page 89)	Restores the current record in the dataset to an unmodified state when cached updates are enabled.
	SQLType (see page 91)	Returns the type of query to be executed.
	Undelete (see page 92)	Restores a record deleted from the dataset.
	UpdateStatus (see page 92)	Reports the update status for the current record.

4.1.2.4.1 TIBCustomDataSet.ActivateConnection Method

Pascal

```
procedure ActivateConnection;
```

C++

```
__fastcall ActivateConnection();
```

Description

This is ActivateConnection, a member of class TIBCustomDataSet.

4.1.2.4.2 TIBCustomDataSet.ActivateTransaction Method

Pascal

```
function ActivateTransaction: Boolean;
```

C++

```
__fastcall Boolean ActivateTransaction();
```

Description

This is ActivateTransaction, a member of class TIBCustomDataSet.

4.1.2.4.3 TIBCustomDataSet.AllocRecordBuffer Method

Pascal

```
function AllocRecordBuffer: PChar; override;
```

C++

```
virtual __fastcall const char * AllocRecordBuffer();
```

Description

This is AllocRecordBuffer, a member of class TIBCustomDataSet.

4.1.2.4.4 TIBCustomDataSet.ApplyUpdates Method

Writes a dataset's pending cached updates to the database.

Pascal

```
procedure ApplyUpdates;
```

C++

```
__fastcall ApplyUpdates();
```

Description

Call ApplyUpdates to write a dataset's pending cached updates to a database. This method passes cached data to the database for storage, but the changes are not committed to the database. An application must explicitly call the TIBTransaction (see page 206) component's Commit method to commit the changes to the database if the write is successful, or call the TIBTransaction (see page 206) component's Rollback method to undo the changes if there is an error.

Note: The preferred method for updating datasets is to call a database component's ApplyUpdates method rather than to call each individual dataset's ApplyUpdates method. The application is responsible for committing or rolling back the transaction.

4.1.2.4.5 TIBCustomDataSet.BookmarkValid Method

Indicates the validity of a specified bookmark.

Pascal

```
function BookmarkValid(Bookmark: TBookmark): Boolean; override;
```

C++

```
virtual __fastcall Boolean BookmarkValid(TBookmark Bookmark);
```

Description

This function returns true if the specified bookmark can be used to identify a record in the dataset, false otherwise.

4.1.2.4.6 TIBCustomDataSet.CachedUpdateStatus Method

Returns the status of the cached updates.

Pascal

```
function CachedUpdateStatus: TCachedUpdateStatus;
```

C++

```
__fastcall TCachedUpdateStatus CachedUpdateStatus();
```

Description

Call CachedUpdateStatus to return the cached update status of the current record in the dataset.

4.1.2.4.7 TIBCustomDataSet.CancelUpdates Method

Clears all pending cached updates from the cache.

Pascal

```
procedure CancelUpdates;
```

C++

```
__fastcall CancelUpdates();
```

Description

Call CancelUpdates to clear all pending cached updates from the cache.

When a dataset is closed, or the CachedUpdates (see page 98) property is set to false, CancelUpdates is called automatically.

Note: To undo changes to a single record, call RevertRecord (see page 89).

4.1.2.4.8 TIBCustomDataSet.CheckDatasetClosed Method

Pascal

```
procedure CheckDatasetClosed;
```

C++

```
__fastcall CheckDatasetClosed();
```

Description

This is CheckDatasetClosed, a member of class TIBCustomDataSet.

4.1.2.4.9 TIBCustomDataSet.CheckDatasetOpen Method

Pascal

```
procedure CheckDatasetOpen;
```

C++

```
__fastcall CheckDatasetOpen();
```

Description

This is CheckDatasetOpen, a member of class TIBCustomDataSet.

4.1.2.4.10 TIBCustomDataSet.CheckNotUniDirectional Method

Pascal

```
procedure CheckNotUniDirectional;
```

C++

```
__fastcall CheckNotUniDirectional();
```

Description

This is CheckNotUniDirectional, a member of class TIBCustomDataSet.

4.1.2.4.11 TIBCustomDataSet.ClearCalcFields Method

Pascal

```
procedure ClearCalcFields(Buffer: PChar); override;
```

C++

```
virtual __fastcall ClearCalcFields(const char * Buffer);
```

Description

This is ClearCalcFields, a member of class TIBCustomDataSet.

4.1.2.4.12 TIBCustomDataSet.CompareBookmarks Method

Compares two bookmarks.

Pascal

```
function CompareBookmarks(Bookmark1: TBookmark; Bookmark2: TBookmark): Integer; override;
```

C++

```
virtual __fastcall int CompareBookmarks(TBookmark Bookmark1, TBookmark Bookmark2);
```

Description

Call CompareBookmarks to compare one bookmark to another. CompareBookmarks returns an integer value indicating the difference between the two bookmarks. CompareBookmarks returns a 0 (zero) if the two bookmarks are the same, a -1 (negative one) if bookmark one is less than bookmark two, or a 1 (positive one) if bookmark one is greater than bookmark two. CompareBookmarks also returns a zero if one of the two bookmarks is nil (Delphi) or NULL (C++).

Bookmark1 and Bookmark2 are the two TBookmark values to compare.

4.1.2.4.13 TIBCustomDataSet.ConstraintsStored Method

Pascal

```
function ConstraintsStored: Boolean;
```

C++

```
__fastcall Boolean ConstraintsStored();
```

Description

This is ConstraintsStored, a member of class TIBCustomDataSet.

4.1.2.4.14 TIBCustomDataSet.CreateBlobStream Method

Returns a TIBDSBlobStream (see page 138) object for reading or writing the data in a specified blob field.

Pascal

```
function CreateBlobStream(Field: TField; Mode: TBlobStreamMode): TStream; override;
```

C++

```
virtual __fastcall TStream CreateBlobStream(TField Field, TBlobStreamMode Mode);
```

Description

Call CreateBlobStream to obtain a stream for reading data from or writing data to a binary large object (BLOB) field. The Field parameter must specify a TBlobField component from the Fields property array. The Mode parameter specifies whether the stream will be used for reading, writing, or updating the contents of the field.

Note: The stream that this method returns belongs to the dataset. Do not free the stream: the dataset handles all allocation and freeing of streams that this method creates.

4.1.2.4.15 TIBCustomDataSet.CreateFields Method

Pascal

```
procedure CreateFields; override;
```

C++

```
virtual __fastcall CreateFields();
```

Description

This is CreateFields, a member of class TIBCustomDataSet.

4.1.2.4.16 TIBCustomDataSet.Current Method

Returns the descriptor area that holds information about the current query.

Pascal

```
function Current: TIBXSQLDA;
```

C++

```
__fastcall TIBXSQLDA Current();
```

Description

Most applications do not need to call the Current method. It returns the extended descriptor area for the dataset's SQL statement, which can be used to retrieve properties of the SQL statement.

4.1.2.4.17 TIBCustomDataSet.DeactivateTransaction Method

Pascal

```
procedure DeactivateTransaction;
```

C++

```
__fastcall DeactivateTransaction();
```

Description

This is DeactivateTransaction, a member of class TIBCustomDataSet.

4.1.2.4.18 TIBCustomDataSet.Disconnect Method

Pascal

```
procedure Disconnect; virtual;
```

C++

```
virtual __fastcall Disconnect();
```

Description

This is Disconnect, a member of class TIBCustomDataSet.

4.1.2.4.19 TIBCustomDataSet.DoBeforeDelete Method

Pascal

```
procedure DoBeforeDelete; override;
```

C++

```
virtual __fastcall DoBeforeDelete();
```

Description

This is DoBeforeDelete, a member of class TIBCustomDataSet.

4.1.2.4.20 TIBCustomDataSet.DoBeforeEdit Method

Pascal

```
procedure DoBeforeEdit; override;
```

C++

```
virtual __fastcall DoBeforeEdit();
```

Description

This is DoBeforeEdit, a member of class TIBCustomDataSet.

4.1.2.4.21 TIBCustomDataSet.DoBeforeInsert Method

Pascal

```
procedure DoBeforeInsert; override;
```

C++

```
virtual __fastcall DoBeforeInsert();
```

Description

This is DoBeforeInsert, a member of class TIBCustomDataSet.

4.1.2.4.22 TIBCustomDataSet.DoOnNewRecord Method

Pascal

```
procedure DoOnNewRecord; override;
```

C++

```
virtual __fastcall DoOnNewRecord();
```

Description

This is DoOnNewRecord, a member of class TIBCustomDataSet.

4.1.2.4.23 TIBCustomDataSet.FetchAll Method

Retrieves all records from the current cursor position to the end of the file and stores them locally.

Pascal

```
procedure FetchAll;
```

C++

```
__fastcall FetchAll();
```

Description

Call FetchAll to reduce network traffic when using cached updates. FetchAll calls CheckBrowseMode to post any pending changes, and then retrieves all records from the current cursor position to the end of the file, and stores them locally. Ordinarily when cached updates are enabled, a transaction retrieves only as much data as it needs for return purposes.

Note: Using FetchAll is not always appropriate. For example, when an application accesses a database used by many simultaneous clients and there is a high degree of contention for updating the same records, fetching all records at once may not be advantageous because some fetched records may be changed by other applications. Always weigh the advantages of reduced network traffic against the need for reduced record contention.

4.1.2.4.24 TIBCustomDataSet.FreeRecordBuffer Method

Pascal

```
procedure FreeRecordBuffer(var Buffer: PChar); override;
```

C++

```
virtual __fastcall FreeRecordBuffer(const char * Buffer);
```

Description

This is FreeRecordBuffer, a member of class TIBCustomDataSet.

4.1.2.4.25 TIBCustomDataSet.GetActiveBuf Method

Pascal

```
function GetActiveBuf: PChar;
```

C++

```
__fastcall const char * GetActiveBuf();
```

Description

This is GetActiveBuf, a member of class TIBCustomDataSet.

4.1.2.4.26 TIBCustomDataSet.GetBookmarkData Method

Pascal

```
procedure GetBookmarkData(Buffer: PChar; Data: Pointer); override;
```

C++

```
virtual __fastcall GetBookmarkData(const char * Buffer, void * Data);
```

Description

This is GetBookmarkData, a member of class TIBCustomDataSet.

4.1.2.4.27 TIBCustomDataSet.GetBookmarkFlag Method

Pascal

```
function GetBookmarkFlag(Buffer: PChar): TBookmarkFlag; override;
```

C++

```
virtual __fastcall TBookmarkFlag GetBookmarkFlag(const char * Buffer);
```

Description

This is GetBookmarkFlag, a member of class TIBCustomDataSet.

4.1.2.4.28 TIBCustomDataSet.GetCanModify Method

Pascal

```
function GetCanModify: Boolean; override;
```

C++

```
virtual __fastcall Boolean GetCanModify();
```

Description

This is GetCanModify, a member of class TIBCustomDataSet.

4.1.2.4.29 TIBCustomDataSet.GetCurrentRecord Method

Retrieves the current record into a buffer.

Pascal

```
function GetCurrentRecord(Buffer: PChar): Boolean; override;
```

C++

```
virtual __fastcall Boolean GetCurrentRecord(const char * Buffer);
```

Description

Most applications should not need to call GetCurrentRecord. TDataSet automatically allocates a buffer for the active record.

Call GetCurrentRecord to copy the current record into a buffer allocated by the application. Buffer must be at least as big as the record size indicated by the RecordSize property.

4.1.2.4.30 TIBCustomDataSet.GetDataSource Method

Pascal

```
function GetDataSource: TDataSource; override;
```

C++

```
virtual __fastcall TDataSource GetDataSource();
```

Description

This is GetDataSource, a member of class TIBCustomDataSet.

4.1.2.4.31 TIBCustomDataSet.GetFieldClass Method

Pascal

```
function GetFieldClass(FieldType: TFieldType): TFieldClass; override;
```

C++

```
virtual __fastcall TFieldClass GetFieldClass(TFieldType FieldType);
```

Description

This is GetFieldClass, a member of class TIBCustomDataSet.

4.1.2.4.32 GetFieldData Method

This is the overview for the GetFieldData method overload.

Overload List

	Name	Description
	TIBCustomDataSet.GetFieldData (TField, Pointer) (see page 77)	Reads the field data into a buffer.
	TIBCustomDataSet.GetFieldData (TField, Pointer, Boolean) (see page 77)	Reads the field data into a buffer.

4.1.2.4.32.1 TIBCustomDataSet.GetFieldData Method (TField, Pointer)

Reads the field data into a buffer.

Pascal

```
function GetFieldData(Field: TField; Buffer: Pointer): Boolean; override; overload;
function GetFieldData(Field: TField; Buffer: Pointer; NativeFormat: Boolean): Boolean;
override; overload;
```

C++

```
virtual __fastcall Boolean GetFieldData(TField Field, void * Buffer);
virtual __fastcall Boolean GetFieldData(TField Field, void * Buffer, Boolean NativeFormat);
```

Description

Reads the field data into a buffer.

Delphi syntax:

```
        function GetFieldData(FieldNo: Integer; Buffer: Pointer): Boolean; overload;
    function GetFieldData(Field: TField; Buffer: Pointer): Boolean; overload; override;
    function GetFieldData(Field: TField; Buffer: Pointer; NativeFormat: Boolean): Boolean;
overload; override;
```

C++ syntax:

```
        virtual bool __fastcall GetFieldData(Db::TField* Field, void * Buffer);
    virtual bool __fastcall GetFieldData(Db::TField* Field, void * Buffer, bool NativeFormat);
    virtual bool __fastcall GetFieldData(int FieldNo, void * Buffer);
```

Description

Most applications do not need to call GetFieldData. TField objects call this method to implement their GetData method.

The Field or FieldNo parameter indicates the field whose data should be fetched. Field specifies the component itself, while FieldNo indicates its field number. The Buffer parameter is a memory buffer with sufficient space to accept the value of the field as it exists in the database (unformatted and untranslated). NativeFormat indicates whether the dataset fetches the field in Delphi's native format for the field type. When NativeFormat is false, the dataset must convert the field value to the native type. This allows the field to handle data from different types of datasets (IBX, BDE-based, and so on) in a uniform manner.

GetFieldData returns true if the data was successfully written to Buffer.

4.1.2.4.33 TIBCustomDataSet.GetRecNo Method

Pascal

```
function GetRecNo: Integer; override;
```

C++

```
virtual __fastcall int GetRecNo();
```

Description

This is GetRecNo, a member of class TIBCustomDataSet.

4.1.2.4.34 TIBCustomDataSet.GetRecord Method

Pascal

```
function GetRecord(Buffer: PChar; GetMode: TGetMode; DoCheck: Boolean): TGetResult; override;
```

C++

```
virtual __fastcall TGetResult GetRecord(const char * Buffer, TGetMode GetMode, Boolean
DoCheck);
```

Description

This is GetRecord, a member of class TIBCustomDataSet.

4.1.2.4.35 TIBCustomDataSet.GetRecordCount Method

Pascal

```
function GetRecordCount: Integer; override;
```

C++

```
virtual __fastcall int GetRecordCount();
```

Description

This is GetRecordCount, a member of class TIBCustomDataSet.

4.1.2.4.36 TIBCustomDataSet.GetRecordSize Method

Pascal

```
function GetRecordSize: Word; override;
```

C++

```
virtual __fastcall Word GetRecordSize();
```

Description

This is GetRecordSize, a member of class TIBCustomDataSet.

4.1.2.4.37 TIBCustomDataSet.InitRecord Method

Pascal

```
procedure InitRecord(Buffer: PChar); override;
```

C++

```
virtual __fastcall InitRecord(const char * Buffer);
```

Description

This is InitRecord, a member of class TIBCustomDataSet.

4.1.2.4.38 TIBCustomDataSet.InternalAddRecord Method

Pascal

```
procedure InternalAddRecord(Buffer: Pointer; Append: Boolean); override;
```

C++

```
virtual __fastcall InternalAddRecord(void * Buffer, Boolean Append);
```

Description

This is InternalAddRecord, a member of class TIBCustomDataSet.

4.1.2.4.39 TIBCustomDataSet.InternalBatchInput Method

Pascal

```
procedure InternalBatchInput(InputObject: TIBBatchInput); virtual;
```

C++

```
virtual __fastcall InternalBatchInput(TIBBatchInput InputObject);
```

Description

This is InternalBatchInput, a member of class TIBCustomDataSet.

4.1.2.4.40 TIBCustomDataSet.InternalBatchOutput Method

Pascal

```
procedure InternalBatchOutput(OutputObject: TIBBatchOutput); virtual;
```

C++

```
virtual __fastcall InternalBatchOutput(TIBBatchOutput OutputObject);
```

Description

This is InternalBatchOutput, a member of class TIBCustomDataSet.

4.1.2.4.41 TIBCustomDataSet.InternalCancel Method

Pascal

```
procedure InternalCancel; override;
```

C++

```
virtual __fastcall InternalCancel();
```

Description

This is InternalCancel, a member of class TIBCustomDataSet.

4.1.2.4.42 TIBCustomDataSet.InternalClose Method

Pascal

```
procedure InternalClose; override;
```

C++

```
virtual __fastcall InternalClose();
```

Description

This is InternalClose, a member of class TIBCustomDataSet.

4.1.2.4.43 TIBCustomDataSet.InternalDelete Method

Pascal

```
procedure InternalDelete; override;
```

C++

```
virtual __fastcall InternalDelete();
```

Description

This is InternalDelete, a member of class TIBCustomDataSet.

4.1.2.4.44 TIBCustomDataSet.InternalExecQuery Method

Pascal

```
procedure InternalExecQuery; virtual;
```

C++

```
virtual __fastcall InternalExecQuery();
```

Description

This is InternalExecQuery, a member of class TIBCustomDataSet.

4.1.2.4.45 TIBCustomDataSet.InternalFirst Method

Pascal

```
procedure InternalFirst; override;
```

C++

```
virtual __fastcall InternalFirst();
```

Description

This is InternalFirst, a member of class TIBCustomDataSet.

4.1.2.4.46 TIBCustomDataSet.InternalGotoBookmark Method

Pascal

```
procedure InternalGotoBookmark(Bookmark: Pointer); override;
```

C++

```
virtual __fastcall InternalGotoBookmark(void * Bookmark);
```

Description

This is InternalGotoBookmark, a member of class TIBCustomDataSet.

4.1.2.4.47 TIBCustomDataSet.InternalHandleException Method

Pascal

```
procedure InternalHandleException; override;
```

C++

```
virtual __fastcall InternalHandleException();
```

Description

This is InternalHandleException, a member of class TIBCustomDataSet.

4.1.2.4.48 TIBCustomDataSet.InternalInitFieldDefs Method

Pascal

```
procedure InternalInitFieldDefs; override;
```

C++

```
virtual __fastcall InternalInitFieldDefs();
```

Description

This is InternalInitFieldDefs, a member of class TIBCustomDataSet.

4.1.2.4.49 TIBCustomDataSet.InternalInitRecord Method

Pascal

```
procedure InternalInitRecord(Buffer: PChar); override;
```

C++

```
virtual __fastcall InternalInitRecord(const char * Buffer);
```

Description

This is InternalInitRecord, a member of class TIBCustomDataSet.

4.1.2.4.50 TIBCustomDataSet.InternalInsert Method

Pascal

```
procedure InternalInsert; override;
```

C++

```
virtual __fastcall InternalInsert();
```

Description

This is InternalInsert, a member of class TIBCustomDataSet.

4.1.2.4.51 TIBCustomDataSet.InternalLast Method

Pascal

```
procedure InternalLast; override;
```

C++

```
virtual __fastcall InternalLast();
```

Description

This is InternalLast, a member of class TIBCustomDataSet.

4.1.2.4.52 TIBCustomDataSet.InternalOpen Method

Pascal

```
procedure InternalOpen; override;
```

C++

```
virtual __fastcall InternalOpen();
```

Description

This is InternalOpen, a member of class TIBCustomDataSet.

4.1.2.4.53 TIBCustomDataSet.InternalPost Method

Pascal

```
procedure InternalPost; override;
```

C++

```
virtual __fastcall InternalPost();
```

Description

This is InternalPost, a member of class TIBCustomDataSet.

4.1.2.4.54 TIBCustomDataSet.InternalPrepare Method

Pascal

```
procedure InternalPrepare; virtual;
```

C++

```
virtual __fastcall InternalPrepare();
```

Description

This is InternalPrepare, a member of class TIBCustomDataSet.

4.1.2.4.55 TIBCustomDataSet.InternalRefresh Method

Pascal

```
procedure InternalRefresh; override;
```

C++

```
virtual __fastcall InternalRefresh();
```

Description

This is InternalRefresh, a member of class TIBCustomDataSet.

4.1.2.4.56 TIBCustomDataSet.InternalRefreshRow Method

Pascal

```
procedure InternalRefreshRow; virtual;
```

C++

```
virtual __fastcall InternalRefreshRow();
```

Description

This is InternalRefreshRow, a member of class TIBCustomDataSet.

4.1.2.4.57 TIBCustomDataSet.InternalSetParamsFromCursor Method

Pascal

```
procedure InternalSetParamsFromCursor; virtual;
```

C++

```
virtual __fastcall InternalSetParamsFromCursor();
```

Description

This is InternalSetParamsFromCursor, a member of class TIBCustomDataSet.

4.1.2.4.58 TIBCustomDataSet.InternalSetToRecord Method

Pascal

```
procedure InternalSetToRecord(Buffer: PChar); override;
```

C++

```
virtual __fastcall InternalSetToRecord(const char * Buffer);
```

Description

This is InternalSetToRecord, a member of class TIBCustomDataSet.

4.1.2.4.59 TIBCustomDataSet.InternalUnPrepare Method

Pascal

```
procedure InternalUnPrepare; virtual;
```

C++

```
virtual __fastcall InternalUnPrepare();
```

Description

This is InternalUnPrepare, a member of class TIBCustomDataSet.

4.1.2.4.60 TIBCustomDataSet.IsCursorOpen Method

Pascal

```
function IsCursorOpen: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsCursorOpen();
```

Description

This is IsCursorOpen, a member of class TIBCustomDataSet.

4.1.2.4.61 TIBCustomDataSet.IsSequenced Method

IsSequenced indicates whether sequence numbers are available for database records.

Pascal

```
function IsSequenced: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsSequenced();
```

Description

IsSequenced indicates whether sequence numbers are available for database records.

Delphi syntax:

```
function IsSequenced: Boolean; override;
```

C++ syntax:

```
virtual bool __fastcall IsSequenced(void);
```

Description

Call IsSequenced to determine whether database records can be located by sequence numbers. When IsSequenced is true, the dataset can navigate directly to a specific record by setting the RecNo property. If IsSequenced is false, the only way to navigate to a specific record is to start at the beginning and count records.

4.1.2.4.62 TIBCustomDataSet.Loaded Method

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

This is Loaded, a member of class TIBCustomDataSet.

4.1.2.4.63 TIBCustomDataSet.Locate Method

Searches the dataset for a specified record and makes that record the current record.

Pascal

```
function Locate(const KeyFields: string; const KeyValues: Variant; Options: TLocateOptions):
Boolean; override;
```

C++

```
virtual __fastcall Boolean Locate(const UnicodeString KeyFields, const Variant KeyValues,
TLocateOptions Options);
```

Description

Call Locate to search a dataset for a specific record and position the cursor on it.

KeyFields is a string containing a semicolon-delimited list of field names on which to search.

KeyValues is a variant that specifies the values to match in the key fields. If KeyFields lists a single field, KeyValues specifies the value for that field on the desired record. To specify multiple search values, pass a variant array as KeyValues, or construct a variant array on the fly using the VarArrayOf routine. For example:

```
        with CustTable do
    Locate('Company;Contact;Phone', VarArrayOf(['Sight Diver', 'P', '408-431-1000']),
[loPartialKey]);
       TLocateOptions Opts;
    Opts.Clear();
    Opts << loPartialKey;
```

```
Variant locvalues[3];
locvalues[0] = Variant("Sight Diver");
locvalues[1] = Variant("P");
locvalues[2] = Variant("408-431-1000");
CustTable->Locate("Company;Contact;Phone", VarArrayOf(locvalues, 2), Opts);
```

Options is a set that optionally specifies additional search latitude when searching on string fields. If Options contains the loCaseInsensitive setting, then Locate ignores case when matching fields. If Options contains the loPartialKey setting, then Locate allows partial-string matching on strings in KeyValues. If Options is an empty set, or if KeyFields does not include any string fields, Options is ignored.

Locate returns true if it finds a matching record, and makes that record the current one. Otherwise Locate returns false.

Locate uses the fastest possible method to locate matching records. If the search fields in KeyFields are indexed and the index is compatible with the specified search options, Locate uses the index. Otherwise Locate creates a filter for the search.

4.1.2.4.64 TIBCustomDataSet.LocateNext Method

Searches the dataset for the record after a specified record and makes that record the current record.

Pascal

```
function LocateNext(const KeyFields: string; const KeyValues: Variant; Options:
TLocateOptions): Boolean;
```

C++

```
__fastcall Boolean LocateNext(const UnicodeString KeyFields, const Variant KeyValues,
TLocateOptions Options);
```

Description

Call LocateNext to search a dataset for a record after the current cursor position

KeyFields is a string containing a semicolon-delimited list of field names on which to search.

KeyValues is a variant that specifies the values to match in the key fields. If KeyFields lists a single field, KeyValues specifies the value for that field on the desired record. To specify multiple search values, pass a variant array as KeyValues, or construct a variant array on the fly using the VarArrayOf routine.

The Options parameter is a set that optionally specifies additional search latitude when searching on string fields. If Options contains the loCaseInsensitive setting, then LocateNext ignores case when matching fields. If Options contains the loPartialKey setting, then LocateNext allows partial-string matching on strings in KeyValues. If Options is an empty set, or if KeyFields does not include any string fields, Options is ignored.

LocateNext returns true if it finds a matching record, and makes that record the current one. Otherwise LocateNext returns false.

LocateNext uses the fastest possible method to locate matching records. If the search fields in KeyFields are indexed and the index is compatible with the specified search options, LocateNext uses the index. Otherwise LocateNext creates a filter for the search.

4.1.2.4.65 TIBCustomDataSet.Lookup Method

Retrieves field values from a record that matches specified search values.

Pascal

```
function Lookup(const KeyFields: string; const KeyValues: Variant; const ResultFields:
string): Variant; override;
```

C++

```
virtual __fastcall Variant Lookup(const UnicodeString KeyFields, const Variant KeyValues,
```

```
const UnicodeString ResultFields);
```

Description

Call Lookup to retrieve values for specified fields from a record that matches search criteria. KeyFields is a string containing a semicolon-delimited list of field names on which to search.

KeyValues is a variant array containing the values to match in the key fields. To specify multiple search values, pass KeyValues as a variant array as an argument, or construct a variant array on the fly using the VarArrayOf routine.

ResultFields is a string containing a semicolon-delimited list of field names whose values should be returned from the matching record.

Lookup returns a variant array containing the values from the fields specified in ResultFields.

If the search fields in KeyFields are indexed, Lookup uses the index. Otherwise Lookup creates a filter for the search.

4.1.2.4.66 TIBCustomDataSet.Post Method

Writes a modified record to the database server.

Pascal

```
procedure Post; override;
```

C++

```
virtual __fastcall Post();
```

Description

Post writes any changes to the current record to the database server. Dataset methods that change the dataset state, such as Edit, Insert, or Append, or that move from one record to another, such as First, Last, Next, and Prior automatically call Post.

Before posting the current record, Post checks the GeneratorField (see page 100) property to determine whether it must generate any field values before posting the record. After changing the generated field, if necessary, Post triggers a BeforePost (see page 97) event, writes the modified record to the server, and then triggers an AfterPost (see page 95) event.

4.1.2.4.67 TIBCustomDataSet.PSEndTransaction Method

Pascal

```
procedure PSEndTransaction(Commit: Boolean); override;
```

C++

```
virtual __fastcall PSEndTransaction(Boolean Commit);
```

Description

This is PSEndTransaction, a member of class TIBCustomDataSet.

4.1.2.4.68 TIBCustomDataSet.PSExecuteStatement Method

Pascal

```
function PSExecuteStatement(const ASQL: string; AParams: TParams; ResultSet: Pointer = nil):
Integer; override;
```

C++

```
virtual __fastcall int PSExecuteStatement(const UnicodeString ASQL, TParams AParams, void *
ResultSet = nil);
```

Description

This is PSExecuteStatement, a member of class TIBCustomDataSet.

4.1.2.4.69 TIBCustomDataSet.PSGetQuoteChar Method

Pascal

```
function PSGetQuoteChar: string; override;
```

C++

```
virtual __fastcall UnicodeString PSGetQuoteChar();
```

Description

This is PSGetQuoteChar, a member of class TIBCustomDataSet.

4.1.2.4.70 TIBCustomDataSet.PSGetTableName Method

Pascal

```
function PSGetTableName: string; override;
```

C++

```
virtual __fastcall UnicodeString PSGetTableName();
```

Description

This is PSGetTableName, a member of class TIBCustomDataSet.

4.1.2.4.71 TIBCustomDataSet.PSGetUpdateException Method

Pascal

```
function PSGetUpdateException(E: Exception; Prev: EUpdateError): EUpdateError; override;
```

C++

```
virtual __fastcall EUpdateError PSGetUpdateException(Exception E, EUpdateError Prev);
```

Description

This is PSGetUpdateException, a member of class TIBCustomDataSet.

4.1.2.4.72 TIBCustomDataSet.PSInTransaction Method

Pascal

```
function PSInTransaction: Boolean; override;
```

C++

```
virtual __fastcall Boolean PSInTransaction();
```

Description

This is PSInTransaction, a member of class TIBCustomDataSet.

4.1.2.4.73 TIBCustomDataSet.PSIsSQLBased Method

Pascal

```
function PSIsSQLBased: Boolean; override;
```

C++

```
virtual __fastcall Boolean PSIsSQLBased();
```

Description

This is PSIsSQLBased, a member of class TIBCustomDataSet.

4.1.2.4.74 TIBCustomDataSet.PSIsSQLSupported Method

Pascal

```
function PSIsSQLSupported: Boolean; override;
```

C++

```
virtual __fastcall Boolean PSIsSQLSupported();
```

Description

This is PSIsSQLSupported, a member of class TIBCustomDataSet.

4.1.2.4.75 TIBCustomDataSet.PSReset Method

Pascal

```
procedure PSReset; override;
```

C++

```
virtual __fastcall PSReset();
```

Description

This is PSReset, a member of class TIBCustomDataSet.

4.1.2.4.76 TIBCustomDataSet.PSStartTransaction Method

Pascal

```
procedure PSStartTransaction; override;
```

C++

```
virtual __fastcall PSStartTransaction();
```

Description

This is PSStartTransaction, a member of class TIBCustomDataSet.

4.1.2.4.77 TIBCustomDataSet.PSUpdateRecord Method

Pascal

```
function PSUpdateRecord(UpdateKind: TUpdateKind; Delta: TDataSet): Boolean; override;
```

C++

```
virtual __fastcall Boolean PSUpdateRecord(TUpdateKind UpdateKind, TDataSet Delta);
```

Description

This is PSUpdateRecord, a member of class TIBCustomDataSet.

4.1.2.4.78 TIBCustomDataSet.RecordModified Method

Sets the record to modified or unmodified.

Pascal

```
procedure RecordModified(Value: Boolean);
```

C++

```
__fastcall RecordModified(Boolean Value);
```

Description

Call RecordModified to mark a record as modified or unmodified.

Note: This method is for internal use.

4.1.2.4.79 TIBCustomDataSet.ReQuery Method

Pascal

```
procedure ReQuery;
```

C++

```
__fastcall ReQuery();
```

Description

This is ReQuery, a member of class TIBCustomDataSet.

4.1.2.4.80 TIBCustomDataSet.RevertRecord Method

Restores the current record in the dataset to an unmodified state when cached updates are enabled.

Pascal

```
procedure RevertRecord;
```

C++

```
__fastcall RevertRecord();
```

Description

Call RevertRecord to undo changes made to the current record when cached updates are enabled.

Note: To undo all changes to all pending updates in the cache, call CancelUpdates (see page 70).

4.1.2.4.81 TIBCustomDataSet.SetActive Method

Pascal

```
procedure SetActive(Value: Boolean); override;
```

C++

```
virtual __fastcall SetActive(Boolean Value);
```

Description

This is SetActive, a member of class TIBCustomDataSet.

4.1.2.4.82 TIBCustomDataSet.SetBookmarkData Method

Pascal

```
procedure SetBookmarkData(Buffer: PChar; Data: Pointer); override;
```

C++

```
virtual __fastcall SetBookmarkData(const char * Buffer, void * Data);
```

Description

This is SetBookmarkData, a member of class TIBCustomDataSet.

4.1.2.4.83 TIBCustomDataSet.SetBookmarkFlag Method

Pascal

```
procedure SetBookmarkFlag(Buffer: PChar; Value: TBookmarkFlag); override;
```

C++

```
virtual __fastcall SetBookmarkFlag(const char * Buffer, TBookmarkFlag Value);
```

Description

This is SetBookmarkFlag, a member of class TIBCustomDataSet.

4.1.2.4.84 TIBCustomDataSet.SetCachedUpdates Method

Pascal

```
procedure SetCachedUpdates(Value: Boolean);
```

C++

```
__fastcall SetCachedUpdates(Boolean Value);
```

Description

This is SetCachedUpdates, a member of class TIBCustomDataSet.

4.1.2.4.85 TIBCustomDataSet.SetDataSource Method

Pascal

```
procedure SetDataSource(Value: TDataSource);
```

C++

```
__fastcall SetDataSource(TDataSource Value);
```

Description

This is SetDataSource, a member of class TIBCustomDataSet.

4.1.2.4.86 SetFieldData Method

This is the overview for the SetFieldData method overload.

Overload List

	Name	Description
	TIBCustomDataSet.SetFieldData (TField, Pointer) (see page 91)	This is SetFieldData, a member of class TIBCustomDataSet.
	TIBCustomDataSet.SetFieldData (TField, Pointer, Boolean) (see page 91)	This is SetFieldData, a member of class TIBCustomDataSet.

4.1.2.4.86.1 TIBCustomDataSet.SetFieldData Method (TField, Pointer)

Pascal

```
procedure SetFieldData(Field: TField; Buffer: Pointer); override;
```

C++

```
virtual __fastcall SetFieldData(TField Field, void * Buffer);
```

Description

This is SetFieldData, a member of class TIBCustomDataSet.

4.1.2.4.86.2 TIBCustomDataSet.SetFieldData Method (TField, Pointer, Boolean)

Pascal

```
procedure SetFieldData(Field: TField; Buffer: Pointer; NativeFormat: Boolean); override;
overload;
```

C++

```
virtual __fastcall SetFieldData(TField Field, void * Buffer, Boolean NativeFormat);
```

Description

This is SetFieldData, a member of class TIBCustomDataSet.

4.1.2.4.87 TIBCustomDataSet.SetRecNo Method

Pascal

```
procedure SetRecNo(Value: Integer); override;
```

C++

```
virtual __fastcall SetRecNo(int Value);
```

Description

This is SetRecNo, a member of class TIBCustomDataSet.

4.1.2.4.88 TIBCustomDataSet.SQLType Method

Returns the type of query to be executed.

Pascal

```
function SQLType: TIBSQLTypes;
```

C++

```
__fastcall TIBSQLTypes SQLType();
```

Description

Use the SQLType to determine the type of query to be executed. This is the same as the SQLType property of the query specified by QSelect (see page 104).

4.1.2.4.89 TIBCustomDataSet.Undelete Method

Restores a record deleted from the dataset.

Pascal

```
procedure Undelete;
```

C++

```
__fastcall Undelete();
```

Description

Call Undelete to restore a record deleted or uninserted from the dataset.

4.1.2.4.90 TIBCustomDataSet.UpdateStatus Method

Reports the update status for the current record.

Pascal

```
function UpdateStatus: TUpdateStatus; override;
```

C++

```
virtual __fastcall TUpdateStatus UpdateStatus();
```

Description

Call UpdateStatus to determine the update status for the current record when cached updates are enabled. Update status can change frequently as records are edited, inserted, or deleted. UpdateStatus offers a convenient method for applications to assess the current status before undertaking or completing operations that depend on the update status of records.

4.1.2.5 TIBCustomDataSet Properties

The properties of the TIBCustomDataSet class are listed here.

Protected Properties

	Name	Description
	BufferChunks (see page 98)	Specifies the dataset buffer chunk size.
	CachedUpdates (see page 98)	Specifies whether cached updates are enabled for a dataset.
	DeleteSQL (see page 99)	Holds the SQL statement used to delete rows from the dataset.
	GeneratorField (see page 100)	Specifies whether the dataset should refresh its data every time it posts a record.
	InsertSQL (see page 100)	Holds the SQL statement used to insert rows into the dataset.
R	InternalPrepared (see page 100)	Indicates whether the dataset has prepared its queries for execution.
R	LiveMode (see page 101)	Indicates what types of edits, if any, can be made to the data in the dataset.
	ModifySQL (see page 101)	Provides the ability to access the SQL object encapsulating the ModifySQL statement.
	ParamCheck (see page 102)	Specifies whether the parameter list for a query is regenerated if the SQL property changes at runtime.
R	Params (see page 103)	Specifies the parameters to use for any parameterized queries.
R	QDelete (see page 103)	Provides the ability to directly access the SQL object encapsulating the DeleteSQL (see page 99) statement.

	QInsert (see page 104)	Provides the ability to directly access the SQL object encapsulating the InsertSQL (see page 100) statement.
	QModify (see page 104)	Provides the ability to directly access the SQL object encapsulating the ModifySQL (see page 101) statement.
	QRefresh (see page 104)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL (see page 105) statement.
	QSelect (see page 104)	Provides the ability to directly access the SQL object encapsulating the SelectSQL (see page 105) statement.
	RefreshSQL (see page 105)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.
	SelectSQL (see page 105)	Provides the ability to directly access the SQL object encapsulating the SelectSQL statement.
	SelectStmtHandle (see page 105)	Gets the handle for the SELECT statement.
	SQLParams (see page 106)	Specifies the parameters to use for any parameterized queries.
	StatementType (see page 106)	Returns the statement type of the QSelect (see page 104) query.
	UniDirectional (see page 107)	Determines whether bidirectional cursors are enabled for the dataset.
	UpdateMode (see page 107)	Indicates how records are located when updating records.

Public Properties

	Name	Description
	DBHandle (see page 99)	Specifies the database handle for the dataset.
	Plan (see page 103)	Indicates the PLAN portion of the dataset's SELECT query.
	RowsAffected (see page 105)	Indicates the number of rows operated on by the latest query execution.
	TRHandle (see page 106)	Specifies the transaction handle for the dataset.
	UpdateObject (see page 107)	Specifies the update object component used to update a read-only result set.
	UpdateRecordTypes (see page 108)	Specifies the type of records visible in a dataset when cached updates are enabled.
	UpdatesPending (see page 108)	Indicates whether the cached updates buffer contains records that are not yet applied.

Published Properties

	Name	Description
	AfterCancel (see page 94)	This is AfterCancel, a member of class TIBCustomDataSet.
	AfterClose (see page 94)	This is AfterClose, a member of class TIBCustomDataSet.
	AfterDelete (see page 94)	This is AfterDelete, a member of class TIBCustomDataSet.
	AfterEdit (see page 94)	This is AfterEdit, a member of class TIBCustomDataSet.
	AfterInsert (see page 95)	This is AfterInsert, a member of class TIBCustomDataSet.
	AfterOpen (see page 95)	This is AfterOpen, a member of class TIBCustomDataSet.
	AfterPost (see page 95)	This is AfterPost, a member of class TIBCustomDataSet.
	AfterRefresh (see page 95)	This is AfterRefresh, a member of class TIBCustomDataSet.
	AfterScroll (see page 95)	This is AfterScroll, a member of class TIBCustomDataSet.
	AutoCalcFields (see page 96)	This is AutoCalcFields, a member of class TIBCustomDataSet.
	BeforeCancel (see page 96)	This is BeforeCancel, a member of class TIBCustomDataSet.
	BeforeClose (see page 96)	This is BeforeClose, a member of class TIBCustomDataSet.
	BeforeDelete (see page 96)	This is BeforeDelete, a member of class TIBCustomDataSet.
	BeforeEdit (see page 97)	This is BeforeEdit, a member of class TIBCustomDataSet.
	BeforeInsert (see page 97)	This is BeforeInsert, a member of class TIBCustomDataSet.
	BeforeOpen (see page 97)	This is BeforeOpen, a member of class TIBCustomDataSet.
	BeforePost (see page 97)	This is BeforePost, a member of class TIBCustomDataSet.
	BeforeRefresh (see page 97)	This is BeforeRefresh, a member of class TIBCustomDataSet.
	BeforeScroll (see page 98)	This is BeforeScroll, a member of class TIBCustomDataSet.
	Database (see page 99)	Identifies the database component for which this dataset represents one or more tables.
	ForcedRefresh (see page 99)	Specifies whether the dataset should refresh its data every time it posts a record.
	ObjectView (see page 101)	This is ObjectView, a member of class TIBCustomDataSet.
	OnCalcFields (see page 101)	This is OnCalcFields, a member of class TIBCustomDataSet.
	OnDeleteError (see page 102)	This is OnDeleteError, a member of class TIBCustomDataSet.

	OnEditError (see page 102)	This is OnEditError, a member of class TIBCustomDataSet.
	OnNewRecord (see page 102)	This is OnNewRecord, a member of class TIBCustomDataSet.
	OnPostError (see page 102)	This is OnPostError, a member of class TIBCustomDataSet.
	Transaction (see page 106)	Identifies the transaction under which the query executes.

4.1.2.5.1 TIBCustomDataSet.AfterCancel Property

Pascal

```
property AfterCancel;
```

C++

```
__property AfterCancel;
```

Description

This is AfterCancel, a member of class TIBCustomDataSet.

4.1.2.5.2 TIBCustomDataSet.AfterClose Property

Pascal

```
property AfterClose;
```

C++

```
__property AfterClose;
```

Description

This is AfterClose, a member of class TIBCustomDataSet.

4.1.2.5.3 TIBCustomDataSet.AfterDelete Property

Pascal

```
property AfterDelete;
```

C++

```
__property AfterDelete;
```

Description

This is AfterDelete, a member of class TIBCustomDataSet.

4.1.2.5.4 TIBCustomDataSet.AfterEdit Property

Pascal

```
property AfterEdit;
```

C++

```
__property AfterEdit;
```

Description

This is AfterEdit, a member of class TIBCustomDataSet.

4.1.2.5.5 TIBCustomDataSet.AfterInsert Property

Pascal

```
property AfterInsert;
```

C++

```
__property AfterInsert;
```

Description

This is AfterInsert, a member of class TIBCustomDataSet.

4.1.2.5.6 TIBCustomDataSet.AfterOpen Property

Pascal

```
property AfterOpen;
```

C++

```
__property AfterOpen;
```

Description

This is AfterOpen, a member of class TIBCustomDataSet.

4.1.2.5.7 TIBCustomDataSet.AfterPost Property

Pascal

```
property AfterPost;
```

C++

```
__property AfterPost;
```

Description

This is AfterPost, a member of class TIBCustomDataSet.

4.1.2.5.8 TIBCustomDataSet.AfterRefresh Property

Pascal

```
property AfterRefresh;
```

C++

```
__property AfterRefresh;
```

Description

This is AfterRefresh, a member of class TIBCustomDataSet.

4.1.2.5.9 TIBCustomDataSet.AfterScroll Property

Pascal

```
property AfterScroll;
```

C++

```
__property AfterScroll;
```

Description

This is AfterScroll, a member of class TIBCustomDataSet.

4.1.2.5.10 TIBCustomDataSet.AutoCalcFields Property

Pascal

```
property AutoCalcFields;
```

C++

```
__property AutoCalcFields;
```

Description

This is AutoCalcFields, a member of class TIBCustomDataSet.

4.1.2.5.11 TIBCustomDataSet.BeforeCancel Property

Pascal

```
property BeforeCancel;
```

C++

```
__property BeforeCancel;
```

Description

This is BeforeCancel, a member of class TIBCustomDataSet.

4.1.2.5.12 TIBCustomDataSet.BeforeClose Property

Pascal

```
property BeforeClose;
```

C++

```
__property BeforeClose;
```

Description

This is BeforeClose, a member of class TIBCustomDataSet.

4.1.2.5.13 TIBCustomDataSet.BeforeDelete Property

Pascal

```
property BeforeDelete;
```

C++

```
__property BeforeDelete;
```

Description

This is BeforeDelete, a member of class TIBCustomDataSet.

4.1.2.5.14 TIBCustomDataSet.BeforeEdit Property

Pascal

```
property BeforeEdit;
```

C++

```
__property BeforeEdit;
```

Description

This is BeforeEdit, a member of class TIBCustomDataSet.

4.1.2.5.15 TIBCustomDataSet.BeforeInsert Property

Pascal

```
property BeforeInsert;
```

C++

```
__property BeforeInsert;
```

Description

This is BeforeInsert, a member of class TIBCustomDataSet.

4.1.2.5.16 TIBCustomDataSet.BeforeOpen Property

Pascal

```
property BeforeOpen;
```

C++

```
__property BeforeOpen;
```

Description

This is BeforeOpen, a member of class TIBCustomDataSet.

4.1.2.5.17 TIBCustomDataSet.BeforePost Property

Pascal

```
property BeforePost;
```

C++

```
__property BeforePost;
```

Description

This is BeforePost, a member of class TIBCustomDataSet.

4.1.2.5.18 TIBCustomDataSet.BeforeRefresh Property

Pascal

```
property BeforeRefresh;
```

C++

```
__property BeforeRefresh;
```

Description

This is BeforeRefresh, a member of class TIBCustomDataSet.

4.1.2.5.19 TIBCustomDataSet.BeforeScroll Property

Pascal

```
property BeforeScroll;
```

C++

```
__property BeforeScroll;
```

Description

This is BeforeScroll, a member of class TIBCustomDataSet.

4.1.2.5.20 TIBCustomDataSet.BufferChunks Property

Specifies the dataset buffer chunk size.

Pascal

```
property BufferChunks: Integer;
```

C++

```
__property int BufferChunks;
```

Description

Use BufferChunks to get or set the number of records for which the dataset allocates buffer space at any time. When the dataset's buffer is full, trying to fetch an additional record causes the dataset to reallocate the buffer so that it has enough memory to hold an additional BufferChunks records.

Note: When UniDirectional (see page 107) is true, BufferChunks has no meaning.

4.1.2.5.21 TIBCustomDataSet.CachedUpdates Property

Specifies whether cached updates are enabled for a dataset.

Pascal

```
property CachedUpdates: Boolean;
```

C++

```
__property Boolean CachedUpdates;
```

Description

CachedUpdates enables or disables the use of cached updates for a dataset. If CachedUpdates is true, cached updates are enabled. If CachedUpdates is false, cached updates are disabled.

When cached updates are enabled, updates to a dataset (such as posting changes, inserting new records, or deleting records), are stored in an internal cache on the client machine instead of being written directly to the dataset's underlying database tables. When changes are complete, an application writes all cached changes to the database in the context of a single transaction.

Note: **Note**: Instead of using cached updates, applications can obtain the same benefits with greater control by using a client dataset.

4.1.2.5.22 TIBCustomDataSet.Database Property

Identifies the database component for which this dataset represents one or more tables.

Pascal

```
property Database: TIBDatabase;
```

C++

```
__property TIBDatabase Database;
```

Description

Use Database to access the properties, events, and methods of the database component associated with this dataset.

See Also

Creating and Deleting Tables

Using Query-type Datasets

Using Stored Procedure-type Datasets

Using Table Type Datasets

Connecting Directly to a Database Server

4.1.2.5.23 TIBCustomDataSet.DBHandle Property

Specifies the database handle for the dataset.

Pascal

```
property DBHandle: PISC_DB_HANDLE;
```

C++

```
__property PISC_DB_HANDLE DBHandle;
```

Description

Check DBHandle to determine the database handle for the dataset.

4.1.2.5.24 TIBCustomDataSet.DeleteSQL Property

Holds the SQL statement used to delete rows from the dataset.

Pascal

```
property DeleteSQL: TStrings;
```

C++

```
__property TStrings DeleteSQL;
```

Description

Use DeleteSQL to delete rows in the dataset.

4.1.2.5.25 TIBCustomDataSet.ForcedRefresh Property

Specifies whether the dataset should refresh its data every time it posts a record.

Pascal

```
property ForcedRefresh: Boolean;
```

C++

```
__property Boolean ForcedRefresh;
```

Description

Set ForcedRefresh to true to force the dataset to refresh its data every time it posts a record. When ForcedRefresh is false (the default), the data is only refreshed when you call the Refresh method, or when the dataset needs to fetch the value of a default or internally calculated field.

4.1.2.5.26 TIBCustomDataSet.GeneratorField Property

Specifies whether the dataset should refresh its data every time it posts a record.

Pascal

```
property GeneratorField: TIBGeneratorField;
```

C++

```
__property TIBGeneratorField GeneratorField;
```

Description

Use GeneratorField to specify a generator that automatically provides the value for one of the fields in the dataset. Use the properties of the TIBGeneratorField (see page 142) object to indicate which field's value should be set and how that value should be calculated.

Warning: Do not set the value of GeneratorField. To change the value of this property, change the properties of the TIBGeneratorField (see page 142) object that is the value of this property.

4.1.2.5.27 TIBCustomDataSet.InsertSQL Property

Holds the SQL statement used to insert rows into the dataset.

Pascal

```
property InsertSQL: TStrings;
```

C++

```
__property TStrings InsertSQL;
```

Description

Use InsertSQL to insert rows into the dataset.

4.1.2.5.28 TIBCustomDataSet.InternalPrepared Property

Indicates whether the dataset has prepared its queries for execution.

Pascal

```
property InternalPrepared: Boolean;
```

C++

```
__property Boolean InternalPrepared;
```

Description

TIBCustomDataSet (see page 62) uses InternalPrepared internally to determine when it needs to prepare its set of queries for

execution. The dataset checks InternalPrepared before executing a query, and prepares the query if InternalPrepared is false.

4.1.2.5.29 TIBCustomDataSet.LiveMode Property

Indicates what types of edits, if any, can be made to the data in the dataset.

Pascal

```
property LiveMode: TLiveModes;
```

C++

```
__property TLiveModes LiveMode;
```

Description

Read LiveMode to determine what types of edits should be allowed. LiveMode is set automatically to restrict the capabilities of the dataset to reflect the user's rights.

Note: For read-only datasets, such as TIBQuery, LiveMode is always an empty set.

4.1.2.5.30 TIBCustomDataSet.ModifySQL Property

Provides the ability to access the SQL object encapsulating the ModifySQL statement.

Pascal

```
property ModifySQL: TStrings;
```

C++

```
__property TStrings ModifySQL;
```

Description

Use ModifySQL to access the SQL object that encapsulates the ModifySQL statement.

4.1.2.5.31 TIBCustomDataSet.ObjectView Property

Pascal

```
property ObjectView;
```

C++

```
__property ObjectView;
```

Description

This is ObjectView, a member of class TIBCustomDataSet.

4.1.2.5.32 TIBCustomDataSet.OnCalcFields Property

Pascal

```
property OnCalcFields;
```

C++

```
__property OnCalcFields;
```

Description

This is OnCalcFields, a member of class TIBCustomDataSet.

4.1.2.5.33 TIBCustomDataSet.OnDeleteError Property

Pascal

```
property OnDeleteError;
```

C++

```
__property OnDeleteError;
```

Description

This is OnDeleteError, a member of class TIBCustomDataSet.

4.1.2.5.34 TIBCustomDataSet.OnEditError Property

Pascal

```
property OnEditError;
```

C++

```
__property OnEditError;
```

Description

This is OnEditError, a member of class TIBCustomDataSet.

4.1.2.5.35 TIBCustomDataSet.OnNewRecord Property

Pascal

```
property OnNewRecord;
```

C++

```
__property OnNewRecord;
```

Description

This is OnNewRecord, a member of class TIBCustomDataSet.

4.1.2.5.36 TIBCustomDataSet.OnPostError Property

Pascal

```
property OnPostError;
```

C++

```
__property OnPostError;
```

Description

This is OnPostError, a member of class TIBCustomDataSet.

4.1.2.5.37 TIBCustomDataSet.ParamCheck Property

Specifies whether the parameter list for a query is regenerated if the SQL property changes at runtime.

Pascal

```
property ParamCheck: Boolean;
```

C++

```
__property Boolean ParamCheck;
```

Description

Set ParamCheck to specify whether the Params (see page 103) property is cleared and regenerated if an application modifies the SQL in this dataset's query. By default ParamCheck is true, meaning that the Params (see page 103) property is automatically regenerated at runtime. When ParamCheck is true, the proper number of parameters is guaranteed to be generated for the current SQL statement.

This property is useful for data definition language (DDL) statements that contain parameters as part of the DDL statement that are not parameters for the TIBCustomDataSet (see page 62). For example, the DDL statement to create a stored procedure may contain parameter statements that are part of the stored procedure. Set ParamCheck to false to prevent these parameters from being mistaken for parameters of the TIBCustomDataSet (see page 62) executing the DDL statement.

4.1.2.5.38 TIBCustomDataSet.Params Property

Specifies the parameters to use for any parameterized queries.

Pascal

```
property Params: TIBXSQLDA;
```

C++

```
__property TIBXSQLDA Params;
```

Description

Use Params to set up the parameters for the SQL object specified by the QSelect (see page 104) property. Params is the same as the SQLParams (see page 106) property.

4.1.2.5.39 TIBCustomDataSet.Plan Property

Indicates the PLAN portion of the dataset's SELECT query.

Pascal

```
property Plan: String;
```

C++

```
__property UnicodeString Plan;
```

Description

Read the Plan property to determine the PLAN clause in the query specified by QSelect (see page 104).

4.1.2.5.40 TIBCustomDataSet.QDelete Property

Provides the ability to directly access the SQL object encapsulating the DeleteSQL (see page 99) statement.

Pascal

```
property QDelete: TIBSQL;
```

C++

```
__property TIBSQL QDelete;
```

Description

Use QDelete to access the SQL object that encapsulates the Delete SQL statement.

4.1.2.5.41 TIBCustomDataSet.QInsert Property

Provides the ability to directly access the SQL object encapsulating the InsertSQL (see page 100) statement.

Pascal

```
property QInsert: TIBSQL;
```

C++

```
__property TIBSQL QInsert;
```

Description

Use QInsert to access the SQL object that encapsulates the InsertSQL (see page 100) statement.

4.1.2.5.42 TIBCustomDataSet.QModify Property

Provides the ability to directly access the SQL object encapsulating the ModifySQL (see page 101) statement.

Pascal

```
property QModify: TIBSQL;
```

C++

```
__property TIBSQL QModify;
```

Description

Use QModify to access the SQL object that encapsulates the ModifySQL (see page 101) statement.

4.1.2.5.43 TIBCustomDataSet.QRefresh Property

Provides the ability to directly access the SQL object encapsulating the RefreshSQL (see page 105) statement.

Pascal

```
property QRefresh: TIBSQL;
```

C++

```
__property TIBSQL QRefresh;
```

Description

Use QRefresh to access the SQL object that encapsulates the RefreshSQL (see page 105) statement.

4.1.2.5.44 TIBCustomDataSet.QSelect Property

Provides the ability to directly access the SQL object encapsulating the SelectSQL (see page 105) statement.

Pascal

```
property QSelect: TIBSQL;
```

C++

```
__property TIBSQL QSelect;
```

Description

Use QSelect to access the SQL object that encapsulates the SelectSQL (see page 105) statement.

4.1.2.5.45 TIBCustomDataSet.RefreshSQL Property

Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.

Pascal

```
property RefreshSQL: TStrings;
```

C++

```
__property TStrings RefreshSQL;
```

Description

Use RefreshSQL to access the SQL object that encapsulates the RefreshSQL statement.

4.1.2.5.46 TIBCustomDataSet.RowsAffected Property

Indicates the number of rows operated on by the latest query execution.

Pascal

```
property RowsAffected: Integer;
```

C++

```
__property int RowsAffected;
```

Description

When the dataset represents a query that does not return a result set, you can't use the RecordCount property to ascertain the number of records affected by the query. In such cases, read RowsAffected to determine how many rows were updated or deleted by the query when it executed. If no rows were updated or deleted, RowsAffected is zero. RowsAffected is -1 if the query could not be executed due to an error condition. (This latter situation typically follows the throwing of an exception.)

4.1.2.5.47 TIBCustomDataSet.SelectSQL Property

Provides the ability to directly access the SQL object encapsulating the SelectSQL statement.

Pascal

```
property SelectSQL: TStrings;
```

C++

```
__property TStrings SelectSQL;
```

Description

Use SelectSQL to access the SQL object that encapsulates the SelectSQL statement.

4.1.2.5.48 TIBCustomDataSet.SelectStmtHandle Property

Gets the handle for the SELECT statement.

Pascal

```
property SelectStmtHandle: TISC_STMT_HANDLE;
```

C++

```
__property TISC_STMT_HANDLE SelectStmtHandle;
```

Description

Use SelectStmtHandle to get the handle for the SELECT statement.

4.1.2.5.49 TIBCustomDataSet.SQLParams Property

Specifies the parameters to use for any parameterized queries.

Pascal

```
property SQLParams: TIBXSQLDA;
```

C++

```
__property TIBXSQLDA SQLParams;
```

Description

Use SQLParams to set up the parameters for the SQL object specified by the QSelect (see page 104) property. SQLParams is the same as the Params (see page 103) property.

4.1.2.5.50 TIBCustomDataSet.StatementType Property

Returns the statement type of the QSelect (see page 104) query.

Pascal

```
property StatementType: TIBSQLTypes;
```

C++

```
__property TIBSQLTypes StatementType;
```

Description

Use StatementType to determine the statement type of a QSelect (see page 104) query.

4.1.2.5.51 TIBCustomDataSet.Transaction Property

Identifies the transaction under which the query executes.

Pascal

```
property Transaction: TIBTransaction;
```

C++

```
__property TIBTransaction Transaction;
```

Description

Use Transaction to determine the transaction that contains any query this dataset executes. Set the properties of this transaction object to control how the transaction executes, or set the Transaction property to cause this dataset's queries to be included in an existing transaction.

4.1.2.5.52 TIBCustomDataSet.TRHandle Property

Specifies the transaction handle for the dataset.

Pascal

```
property TRHandle: PISC_TR_HANDLE;
```

C++

```
__property PISC_TR_HANDLE TRHandle;
```

Description

Check TRHandle to determine the transaction handle for the dataset.

4.1.2.5.53 TIBCustomDataSet.UniDirectional Property

Determines whether bidirectional cursors are enabled for the dataset.

Pascal

```
property UniDirectional: Boolean;
```

C++

```
__property Boolean UniDirectional;
```

Description

Use UniDirectional to determine whether bidirectional cursors are enabled for this dataset.

Note: Do not confuse the UniDirectional property with the IsUniDirectional property inherited from TDataSet. IsUniDirectional indicates whether the dataset is a dbExpress dataset, which, in addition to using a unidirectional cursor, does not buffer records. UniDirectional is only concerned with whether the cursor permits backward or random navigation.

4.1.2.5.54 TIBCustomDataSet.UpdateMode Property

Indicates how records are located when updating records.

Pascal

```
property UpdateMode: TUpdateMode;
```

C++

```
__property TUpdateMode UpdateMode;
```

Description

UpdateMode is not currently implemented in IB (see page 1) datasets. It is reserved for future use.

4.1.2.5.55 TIBCustomDataSet.UpdateObject Property

Specifies the update object component used to update a read-only result set.

Pascal

```
property UpdateObject: TIBDataSetUpdateObject;
```

C++

```
__property TIBDataSetUpdateObject UpdateObject;
```

Description

Use UpdateObject to specify the TIBDataSetUpdateObject (see page 134) component to use in an application that must be able to update a read-only result set.

In some cases, such as when using TIBQuery, the dataset does not automatically update records. In these cases, UpdateObject can be used to specify a TIBUpdateSQL component that performs updates as a separate transaction that is transparent to the application.

TIBDataSetUpdateObject (see page 134) is especially useful for complicated updates from an OnUpdateRecord (see page 111) event.

4.1.2.5.56 TIBCustomDataSet.UpdateRecordTypes Property

Specifies the type of records visible in a dataset when cached updates are enabled.

Pascal

```
property UpdateRecordTypes: TIBUpdateRecordTypes;
```

C++

```
__property TIBUpdateRecordTypes UpdateRecordTypes;
```

Description

Use UpdateRecordTypes to specify the records that are visible in a dataset when cached updates are enabled. UpdateRecordTypes is a set that can have the following values:

By default, a dataset is created with an UpdateRecordTypes set of cusModified, cusInserted, or cusUnmodified, meaning that all existing, edited, or inserted records are visible to the user.

An application that must cycle through a dataset to undelete records may change UpdateRecordTypes as part of an undelete method, so that deleted records are 'visible' long enough to restore them to their previously undeleted conditions.

Similarly, an application that must cycle through a dataset to uninsert records may change UpdateRecordTypes as part of an uninsert method, so that uninserted records are 'visible' long enough to restore them to their previously inserted conditions.

An application might also use UpdateRecordTypes like a filter to temporarily limit visible records to those added or inserted by the user during the current session.

4.1.2.5.57 TIBCustomDataSet.UpdatesPending Property

Indicates whether the cached updates buffer contains records that are not yet applied.

Pascal

```
property UpdatesPending: Boolean;
```

C++

```
__property Boolean UpdatesPending;
```

Description

Examine UpdatesPending to check the status of the cached updates buffer. If UpdatesPending is true, then there are edited, deleted, or inserted records to apply to the database. If UpdatesPending is false, there are no records in the cache.

4.1.2.6 TIBCustomDataSet Events

The events of the TIBCustomDataSet class are listed here.

Protected Events

	Name	Description
	AfterDatabaseDisconnect (see page 109)	Occurs immediately after the connection to the associated database is closed.
	AfterTransactionEnd (see page 109)	Occurs immediately after a transaction that includes this dataset's query ends.
	BeforeDatabaseDisconnect (see page 109)	Occurs immediately before the connection to the associated database is closed.
	BeforeTransactionEnd (see page 109)	Occurs immediately before a transaction that includes this dataset's query ends.
	DatabaseFree (see page 110)	Occurs when the database associated with this dataset is freed.

	TransactionFree (see page 111)	Occurs when the transaction that includes this dataset's query is freed.

Published Events

	Name	Description
	OnUpdateError (see page 110)	Occurs if an exception is generated when cached updates are applied to a database.
	OnUpdateRecord (see page 111)	Occurs when cached updates are applied to a record.

4.1.2.6.1 TIBCustomDataSet.AfterDatabaseDisconnect Event

Occurs immediately after the connection to the associated database is closed.

Pascal

```
property AfterDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterDatabaseDisconnect;
```

Description

Write an AfterDatabaseDisconnect event handler to respond when the dataset closes its connection to the database specified by the Database (see page 99) property.

4.1.2.6.2 TIBCustomDataSet.AfterTransactionEnd Event

Occurs immediately after a transaction that includes this dataset's query ends.

Pascal

```
property AfterTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterTransactionEnd;
```

Description

Write an AfterTransactionEnd event handler to respond when the transaction specified by the Transaction (see page 106) property ends.

4.1.2.6.3 TIBCustomDataSet.BeforeDatabaseDisconnect Event

Occurs immediately before the connection to the associated database is closed.

Pascal

```
property BeforeDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeDatabaseDisconnect;
```

Description

Write a BeforeDatabaseDisconnect event handler to respond when the dataset is about to close its connection to the database specified by the Database (see page 99) property.

4.1.2.6.4 TIBCustomDataSet.BeforeTransactionEnd Event

Occurs immediately before a transaction that includes this dataset's query ends.

Pascal

```
property BeforeTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeTransactionEnd;
```

Description

Write a BeforeTransactionEnd event handler to respond when the transaction specified by the Transaction (see page 106) property is about to end.

4.1.2.6.5 TIBCustomDataSet.DatabaseFree Event

Occurs when the database associated with this dataset is freed.

Pascal

```
property DatabaseFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent DatabaseFree;
```

Description

Write a DatabaseFree event handler to respond when the database specified by the Database (see page 99) property is freed.

4.1.2.6.6 TIBCustomDataSet.OnUpdateError Event

Occurs if an exception is generated when cached updates are applied to a database.

Pascal

```
property OnUpdateError: TIBUpdateErrorEvent;
```

C++

```
__property TIBUpdateErrorEvent OnUpdateError;
```

Description

Write an OnUpdateError event handler to respond to exceptions generated when cached updates are applied to a database.

Because there is a delay between the time a record is first cached and the time cached updates are applied, there is a possibility that another application may change one or more of the same records in the database before the cached changes can be applied. DataSet is the name of the dataset to which updates are applied.

The error handler can use the OldValue and NewValue properties of TField to evaluate error conditions, and set NewValue to a new value to reapply. In this case, set UpdateAction to uaRetry before exiting.

Note: If a call to ApplyUpdates (see page 69) raises an exception and ApplyUpdates (see page 69) is not called within the context of a try..except (Delphi) or try..catch (C++) block, an error message is returned. If an OnUpdateError handler cannot correct the error condition and leaves UpdateAction set to uaFail, the error message is returned twice. To prevent re-return, set UpdateAction to uaAbort in the error handler.

Warning: **Important:** The code in an OnUpdateError handler must not call any methods that make a different record the current one.

4.1.2.6.7 TIBCustomDataSet.OnUpdateRecord Event

Occurs when cached updates are applied to a record.

Pascal

```
property OnUpdateRecord: TIBUpdateRecordEvent;
```

C++

```
__property TIBUpdateRecordEvent OnUpdateRecord;
```

Description

Write an OnUpdateRecord event handler to process updates that cannot be handled by a single update component, such as implementation of cascading updates, insertions, or deletions. This handler is also useful for applications that require additional control over parameter substitution in update components.

Note: The code in an OnUpdateRecord handler must not call any methods that make a different record the current one.

4.1.2.6.8 TIBCustomDataSet.TransactionFree Event

Occurs when the transaction that includes this dataset's query is freed.

Pascal

```
property TransactionFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent TransactionFree;
```

Description

Write a TransactionFree event handler to respond when the transaction specified by the Transaction (see page 106) property is freed.

4.1.3 TIBDataLink Class

TIBDataLink is a helper class used by data-aware objects to coordinate the actions of TIBDataSet (see page 114) and to respond to data events.

Class Hierarchy

Pascal

```
TIBDataLink = class(TDetailDataLink);
```

C++

```
class TIBDataLink : public TDetailDataLink;
```

File

IBCustomDataSet

Description

Use TIBDataLink or one of its descendants in any data-aware object that implements a DataSource property to represent its link to a dataset or that needs to respond to data events.

The constructor of the data-aware object should call the constructor of TIBDataLink, and initialize any relevant properties. The data-aware object can then link to a TDataSource by using the DataSource property of the TIBDataLink.

Data-aware objects that link to a single field in a dataset should use a TFieldDataLink instead.

4.1.3.1 TIBDataLink Members

The following tables list the members exposed by TIBDataLink.

Protected Methods

	Name	Description
	ActiveChanged (see page 113)	This is ActiveChanged, a member of class TIBDataLink.
	CheckBrowseMode (see page 113)	This is CheckBrowseMode, a member of class TIBDataLink.
	GetDetailDataSet (see page 113)	This is GetDetailDataSet, a member of class TIBDataLink.
	RecordChanged (see page 114)	This is RecordChanged, a member of class TIBDataLink.

Public Methods

	Name	Description
	Create (see page 112)	Creates an instance of TIBDataLink (see page 111).
	Destroy (see page 112)	Destroys an instance of TIBDataLink (see page 111).

4.1.3.2 TIBDataLink.Create Constructor

Creates an instance of TIBDataLink (see page 111).

Pascal

```
constructor Create(ADataSet: TIBCustomDataSet);
```

C++

```
__fastcall TIBDataLink(TIBCustomDataSet ADataSet);
```

Description

Create is called from the constructor of any data-aware object that uses a TIBDataLink (see page 111) to implement its DataSource property.

After calling the inherited constructor, Create initializes the BufferCount property to 1. Data-aware objects that use a TIBDataLink (see page 111) object to manage their link to a DataSource should change the BufferCount property to the number or records they represent, after calling the inherited constructor.

4.1.3.3 TIBDataLink.Destroy Destructor

Destroys an instance of TIBDataLink (see page 111).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBDataLink();
```

Description

Do not call Destroy directly in an application. Instead, call the Free method. Free verifies that the TIBDataLink (see page 111) object is not already freed and only then calls Destroy.

Before calling the inherited destructor, Destroy removes any reference to the TIBDataLink (see page 111) from the data source object.

The TIBDataLink (see page 111) object should be destroyed in the destructor of its Owner, where that Owner calls Create (see page 112) from its constructor.

4.1.3.4 TIBDataLink Methods

The methods of the TIBDataLink class are listed here.

Protected Methods

	Name	Description
	ActiveChanged (see page 113)	This is ActiveChanged, a member of class TIBDataLink.
	CheckBrowseMode (see page 113)	This is CheckBrowseMode, a member of class TIBDataLink.
	GetDetailDataSet (see page 113)	This is GetDetailDataSet, a member of class TIBDataLink.
	RecordChanged (see page 114)	This is RecordChanged, a member of class TIBDataLink.

4.1.3.4.1 TIBDataLink.ActiveChanged Method

Pascal

```
procedure ActiveChanged; override;
```

C++

```
virtual __fastcall ActiveChanged();
```

Description

This is ActiveChanged, a member of class TIBDataLink.

4.1.3.4.2 TIBDataLink.CheckBrowseMode Method

Pascal

```
procedure CheckBrowseMode; override;
```

C++

```
virtual __fastcall CheckBrowseMode();
```

Description

This is CheckBrowseMode, a member of class TIBDataLink.

4.1.3.4.3 TIBDataLink.GetDetailDataSet Method

Pascal

```
function GetDetailDataSet: TDataSet; override;
```

C++

```
virtual __fastcall TDataSet GetDetailDataSet();
```

Description

This is GetDetailDataSet, a member of class TIBDataLink.

4.1.3.4.4 TIBDataLink.RecordChanged Method

Pascal

```
procedure RecordChanged(Field: TField); override;
```

C++

```
virtual __fastcall RecordChanged(TField Field);
```

Description

This is RecordChanged, a member of class TIBDataLink.

4.1.4 TIBDataSet Class

TIBDataSet executes InterBase SQL statements.

Class Hierarchy

Pascal

```
TIBDataSet = class(TIBCustomDataSet);
```

C++

```
class TIBDataSet : public TIBCustomDataSet;
```

File

IBCustomDataSet

Description

Use TIBDataSet to execute InterBase SQL statements. Each TIBDataSet component has properties for a statement to fetch data (typically an SQL SELECT statement), a statement to modify those records, one for inserting records, one for deleting records, and one to refresh the dataset.

TIBDataSet buffers the result set, making it completely scrollable. Since TIBDataSet is a descendant of TDataSet, it works well with all data-aware components.

See Also

Types of Datasets

Getting Started with InterBase Express

4.1.4.1 TIBDataSet Members

The following tables list the members exposed by TIBDataSet.

Protected Methods

	Name	Description
	ActivateConnection (see page 69)	This is ActivateConnection, a member of class TIBCustomDataSet.
	ActivateTransaction (see page 69)	This is ActivateTransaction, a member of class TIBCustomDataSet.
V	AllocRecordBuffer (see page 69)	This is AllocRecordBuffer, a member of class TIBCustomDataSet.
	CheckDatasetClosed (see page 71)	This is CheckDatasetClosed, a member of class TIBCustomDataSet.

	CheckDatasetOpen (see page 71)	This is CheckDatasetOpen, a member of class TIBCustomDataSet.
	CheckNotUniDirectional (see page 71)	This is CheckNotUniDirectional, a member of class TIBCustomDataSet.
V	ClearCalcFields (see page 71)	This is ClearCalcFields, a member of class TIBCustomDataSet.
	ConstraintsStored (see page 72)	This is ConstraintsStored, a member of class TIBCustomDataSet.
V	CreateFields (see page 72)	This is CreateFields, a member of class TIBCustomDataSet.
	DeactivateTransaction (see page 73)	This is DeactivateTransaction, a member of class TIBCustomDataSet.
V	Disconnect (see page 73)	This is Disconnect, a member of class TIBCustomDataSet.
V	DoBeforeDelete (see page 73)	This is DoBeforeDelete, a member of class TIBCustomDataSet.
V	DoBeforeEdit (see page 73)	This is DoBeforeEdit, a member of class TIBCustomDataSet.
V	DoBeforeInsert (see page 74)	This is DoBeforeInsert, a member of class TIBCustomDataSet.
V	DoOnNewRecord (see page 74)	This is DoOnNewRecord, a member of class TIBCustomDataSet.
V	FreeRecordBuffer (see page 75)	This is FreeRecordBuffer, a member of class TIBCustomDataSet.
	GetActiveBuf (see page 75)	This is GetActiveBuf, a member of class TIBCustomDataSet.
V	GetBookmarkData (see page 75)	This is GetBookmarkData, a member of class TIBCustomDataSet.
V	GetBookmarkFlag (see page 75)	This is GetBookmarkFlag, a member of class TIBCustomDataSet.
V	GetCanModify (see page 75)	This is GetCanModify, a member of class TIBCustomDataSet.
V	GetDataSource (see page 76)	This is GetDataSource, a member of class TIBCustomDataSet.
V	GetFieldClass (see page 76)	This is GetFieldClass, a member of class TIBCustomDataSet.
V	GetRecNo (see page 77)	This is GetRecNo, a member of class TIBCustomDataSet.
V	GetRecord (see page 77)	This is GetRecord, a member of class TIBCustomDataSet.
V	GetRecordCount (see page 78)	This is GetRecordCount, a member of class TIBCustomDataSet.
V	GetRecordSize (see page 78)	This is GetRecordSize, a member of class TIBCustomDataSet.
V	InitRecord (see page 78)	This is InitRecord, a member of class TIBCustomDataSet.
V	InternalAddRecord (see page 78)	This is InternalAddRecord, a member of class TIBCustomDataSet.
V	InternalBatchInput (see page 79)	This is InternalBatchInput, a member of class TIBCustomDataSet.
V	InternalBatchOutput (see page 79)	This is InternalBatchOutput, a member of class TIBCustomDataSet.
V	InternalCancel (see page 79)	This is InternalCancel, a member of class TIBCustomDataSet.
V	InternalClose (see page 79)	This is InternalClose, a member of class TIBCustomDataSet.
V	InternalDelete (see page 79)	This is InternalDelete, a member of class TIBCustomDataSet.
V	InternalExecQuery (see page 80)	This is InternalExecQuery, a member of class TIBCustomDataSet.
V	InternalFirst (see page 80)	This is InternalFirst, a member of class TIBCustomDataSet.
V	InternalGotoBookmark (see page 80)	This is InternalGotoBookmark, a member of class TIBCustomDataSet.
V	InternalHandleException (see page 80)	This is InternalHandleException, a member of class TIBCustomDataSet.
V	InternalInitFieldDefs (see page 81)	This is InternalInitFieldDefs, a member of class TIBCustomDataSet.
V	InternalInitRecord (see page 81)	This is InternalInitRecord, a member of class TIBCustomDataSet.
V	InternalInsert (see page 81)	This is InternalInsert, a member of class TIBCustomDataSet.
V	InternalLast (see page 81)	This is InternalLast, a member of class TIBCustomDataSet.
V	InternalOpen (see page 119)	This is InternalOpen, a member of class TIBDataSet.
V	InternalPost (see page 82)	This is InternalPost, a member of class TIBCustomDataSet.
V	InternalPrepare (see page 82)	This is InternalPrepare, a member of class TIBCustomDataSet.
V	InternalRefresh (see page 82)	This is InternalRefresh, a member of class TIBCustomDataSet.
V	InternalRefreshRow (see page 82)	This is InternalRefreshRow, a member of class TIBCustomDataSet.
V	InternalSetParamsFromCursor (see page 83)	This is InternalSetParamsFromCursor, a member of class TIBCustomDataSet.
V	InternalSetToRecord (see page 83)	This is InternalSetToRecord, a member of class TIBCustomDataSet.
V	InternalUnPrepare (see page 83)	This is InternalUnPrepare, a member of class TIBCustomDataSet.
V	IsCursorOpen (see page 83)	This is IsCursorOpen, a member of class TIBCustomDataSet.
V	Loaded (see page 84)	This is Loaded, a member of class TIBCustomDataSet.
V	PSEndTransaction (see page 86)	This is PSEndTransaction, a member of class TIBCustomDataSet.
V	PSExecuteStatement (see page 86)	This is PSExecuteStatement, a member of class TIBCustomDataSet.
V	PSGetQuoteChar (see page 87)	This is PSGetQuoteChar, a member of class TIBCustomDataSet.
V	PSGetTableName (see page 87)	This is PSGetTableName, a member of class TIBCustomDataSet.
V	PSGetUpdateException (see page 87)	This is PSGetUpdateException, a member of class TIBCustomDataSet.
V	PSInTransaction (see page 87)	This is PSInTransaction, a member of class TIBCustomDataSet.
V	PSIsSQLBased (see page 88)	This is PSIsSQLBased, a member of class TIBCustomDataSet.

	PSIsSQLSupported (see page 88)	This is PSIsSQLSupported, a member of class TIBCustomDataSet.
	PSReset (see page 88)	This is PSReset, a member of class TIBCustomDataSet.
	PSSetCommandText (see page 120)	This is PSSetCommandText, a member of class TIBDataSet.
	PSStartTransaction (see page 88)	This is PSStartTransaction, a member of class TIBCustomDataSet.
	PSUpdateRecord (see page 88)	This is PSUpdateRecord, a member of class TIBCustomDataSet.
	ReQuery (see page 89)	This is ReQuery, a member of class TIBCustomDataSet.
	SetActive (see page 89)	This is SetActive, a member of class TIBCustomDataSet.
	SetBookmarkData (see page 90)	This is SetBookmarkData, a member of class TIBCustomDataSet.
	SetBookmarkFlag (see page 90)	This is SetBookmarkFlag, a member of class TIBCustomDataSet.
	SetCachedUpdates (see page 90)	This is SetCachedUpdates, a member of class TIBCustomDataSet.
	SetDataSource (see page 90)	This is SetDataSource, a member of class TIBCustomDataSet.
	SetFieldData (see page 90)	This is the overview for the SetFieldData method overload.
	SetFiltered (see page 120)	This is SetFiltered, a member of class TIBDataSet.
	SetRecNo (see page 91)	This is SetRecNo, a member of class TIBCustomDataSet.

Protected Properties

	Name	Description
	InternalPrepared (see page 100)	Indicates whether the dataset has prepared its queries for execution.
	QDelete (see page 103)	Provides the ability to directly access the SQL object encapsulating the DeleteSQL (see page 99) statement.
	QInsert (see page 104)	Provides the ability to directly access the SQL object encapsulating the InsertSQL (see page 100) statement.
	QModify (see page 104)	Provides the ability to directly access the SQL object encapsulating the ModifySQL (see page 101) statement.
	QRefresh (see page 104)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL (see page 105) statement.
	QSelect (see page 104)	Provides the ability to directly access the SQL object encapsulating the SelectSQL (see page 105) statement.
	SQLParams (see page 106)	Specifies the parameters to use for any parameterized queries.
	UpdateMode (see page 107)	Indicates how records are located when updating records.

Public Methods

	Name	Description
	ApplyUpdates (see page 69)	Writes a dataset's pending cached updates to the database.
	BatchInput (see page 119)	Executes the parameterized query in SQL for input in the referenced input object.
	BatchOutput (see page 119)	Outputs the selected query in SQL to the referenced OutputObject.
	BookmarkValid (see page 70)	Indicates the validity of a specified bookmark.
	CachedUpdateStatus (see page 70)	Returns the status of the cached updates.
	CancelUpdates (see page 70)	Clears all pending cached updates from the cache.
	CompareBookmarks (see page 71)	Compares two bookmarks.
	Create (see page 66)	Creates an instance of a TDataSet component.
	CreateBlobStream (see page 72)	Returns a TIBDSBlobStream (see page 138) object for reading or writing the data in a specified blob field.
	Current (see page 73)	Returns the descriptor area that holds information about the current query.
	Destroy (see page 66)	Destroys the instance of a dataset component.
	ExecSQL (see page 119)	Executes the SQL statement that the SelectSQL (see page 131) property specifies when it is not a SELECT statement.
	FetchAll (see page 74)	Retrieves all records from the current cursor position to the end of the file and stores them locally.
	GetCurrentRecord (see page 76)	Retrieves the current record into a buffer.
	GetFieldData (see page 76)	This is the overview for the GetFieldData method overload.
	IsSequenced (see page 83)	IsSequenced indicates whether sequence numbers are available for database records.
	Locate (see page 84)	Searches the dataset for a specified record and makes that record the current record.
	LocateNext (see page 85)	Searches the dataset for the record after a specified record and makes that record the current record.
	Lookup (see page 85)	Retrieves field values from a record that matches specified search values.

	ParamByName (see page 120)	Returns a specified parameter of the SELECT query that fetches this dataset's data.
V	Post (see page 86)	Writes a modified record to the database server.
	Prepare (see page 120)	Prepares all queries in the dataset to be executed.
	RecordModified (see page 89)	Sets the record to modified or unmodified.
	RevertRecord (see page 89)	Restores the current record in the dataset to an unmodified state when cached updates are enabled.
	SQLType (see page 91)	Returns the type of query to be executed.
	Undelete (see page 92)	Restores a record deleted from the dataset.
	UnPrepare (see page 121)	Resets the state of a dataset's internal queries.
V	UpdateStatus (see page 92)	Reports the update status for the current record.

Public Properties

	Name	Description
R	DBHandle (see page 99)	Specifies the database handle for the dataset.
	LiveMode (see page 128)	Indicates what types of edits, if any, can be made to the data in the dataset.
	Params (see page 130)	Provides the ability to specify values for a parameterized query.
R	Plan (see page 103)	Indicates the PLAN portion of the dataset's SELECT query.
R	Prepared (see page 131)	Determines whether or not a set of dataset queries is prepared for execution.
R	RowsAffected (see page 105)	Indicates the number of rows operated on by the latest query execution.
	SelectStmtHandle (see page 131)	Gets the handle for the SELECT statement.
	StatementType (see page 132)	Returns the statement type of the QSelect query.
R	TRHandle (see page 106)	Specifies the transaction handle for the dataset.
	UpdateRecordTypes (see page 108)	Specifies the type of records visible in a dataset when cached updates are enabled.
R	UpdatesPending (see page 108)	Indicates whether the cached updates buffer contains records that are not yet applied.

Published Events

	Name	Description
	AfterDatabaseDisconnect (see page 133)	Occurs immediately after the connection to the associated database is closed.
	AfterTransactionEnd (see page 133)	Occurs immediately after a transaction that includes this dataset's query ends.
	BeforeDatabaseDisconnect (see page 133)	Occurs immediately before the connection to the associated database is closed.
	BeforeTransactionEnd (see page 134)	Occurs immediately before a transaction that includes this dataset's query ends.
	DatabaseFree (see page 134)	Occurs when the database associated with this dataset is freed.
	OnUpdateError (see page 110)	Occurs if an exception is generated when cached updates are applied to a database.
	OnUpdateRecord (see page 111)	Occurs when cached updates are applied to a record.
	TransactionFree (see page 134)	Occurs when the transaction that includes this dataset's query is freed.

Published Properties

	Name	Description
	Active (see page 122)	This is Active, a member of class TIBDataSet.
	AfterCancel (see page 122)	This is AfterCancel, a member of class TIBDataSet.
	AfterClose (see page 122)	This is AfterClose, a member of class TIBDataSet.
	AfterDelete (see page 123)	This is AfterDelete, a member of class TIBDataSet.
	AfterEdit (see page 123)	This is AfterEdit, a member of class TIBDataSet.
	AfterInsert (see page 123)	This is AfterInsert, a member of class TIBDataSet.
	AfterOpen (see page 123)	This is AfterOpen, a member of class TIBDataSet.
	AfterPost (see page 123)	This is AfterPost, a member of class TIBDataSet.
	AfterRefresh (see page 95)	This is AfterRefresh, a member of class TIBCustomDataSet.
	AfterScroll (see page 124)	This is AfterScroll, a member of class TIBDataSet.
	AutoCalcFields (see page 124)	This is AutoCalcFields, a member of class TIBDataSet.
	BeforeCancel (see page 124)	This is BeforeCancel, a member of class TIBDataSet.
	BeforeClose (see page 124)	This is BeforeClose, a member of class TIBDataSet.
	BeforeDelete (see page 125)	This is BeforeDelete, a member of class TIBDataSet.

	BeforeEdit (see page 125)	This is BeforeEdit, a member of class TIBDataSet.
	BeforeInsert (see page 125)	This is BeforeInsert, a member of class TIBDataSet.
	BeforeOpen (see page 125)	This is BeforeOpen, a member of class TIBDataSet.
	BeforePost (see page 125)	This is BeforePost, a member of class TIBDataSet.
	BeforeRefresh (see page 97)	This is BeforeRefresh, a member of class TIBCustomDataSet.
	BeforeScroll (see page 126)	This is BeforeScroll, a member of class TIBDataSet.
	BufferChunks (see page 126)	Specifies the dataset buffer chunk size.
	CachedUpdates (see page 126)	Specifies whether cached updates are enabled for a dataset.
	Database (see page 99)	Identifies the database component for which this dataset represents one or more tables.
	DataSource (see page 127)	This is DataSource, a member of class TIBDataSet.
	DeleteSQL (see page 127)	Holds the SQL statement used to delete rows from the dataset.
	Filtered (see page 127)	Specifies whether or not filtering is active for the dataset.
	ForcedRefresh (see page 99)	Specifies whether the dataset should refresh its data every time it posts a record.
	GeneratorField (see page 127)	Specifies whether the dataset should refresh its data every time it posts a record.
	InsertSQL (see page 128)	Holds the SQL statement used to insert rows into the dataset.
	ModifySQL (see page 128)	Provides the ability to access the SQL object encapsulating the ModifySQL statement.
	ObjectView (see page 101)	This is ObjectView, a member of class TIBCustomDataSet.
	OnCalcFields (see page 129)	This is OnCalcFields, a member of class TIBDataSet.
	OnDeleteError (see page 129)	This is OnDeleteError, a member of class TIBDataSet.
	OnEditError (see page 129)	This is OnEditError, a member of class TIBDataSet.
	OnFilterRecord (see page 129)	This is OnFilterRecord, a member of class TIBDataSet.
	OnNewRecord (see page 129)	This is OnNewRecord, a member of class TIBDataSet.
	OnPostError (see page 130)	This is OnPostError, a member of class TIBDataSet.
	ParamCheck (see page 130)	Specifies whether the parameter list for a query is regenerated if the SQL property changes at runtime.
	RefreshSQL (see page 131)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.
	SelectSQL (see page 131)	Provides the ability to directly access the SQL object encapsulating the SelectSQL statement.
	Transaction (see page 106)	Identifies the transaction under which the query executes.
	UniDirectional (see page 132)	Determines whether bidirectional cursors are enabled for the dataset.
	UpdateObject (see page 132)	Specifies the update object component used to update a read-only result set.

4.1.4.2 TIBDataSet Methods

The methods of the TIBDataSet class are listed here.

Protected Methods

	Name	Description
	InternalOpen (see page 119)	This is InternalOpen, a member of class TIBDataSet.
	PSSetCommandText (see page 120)	This is PSSetCommandText, a member of class TIBDataSet.
	SetFiltered (see page 120)	This is SetFiltered, a member of class TIBDataSet.

Public Methods

	Name	Description
	BatchInput (see page 119)	Executes the parameterized query in SQL for input in the referenced input object.
	BatchOutput (see page 119)	Outputs the selected query in SQL to the referenced OutputObject.
	ExecSQL (see page 119)	Executes the SQL statement that the SelectSQL (see page 131) property specifies when it is not a SELECT statement.
	ParamByName (see page 120)	Returns a specified parameter of the SELECT query that fetches this dataset's data.
	Prepare (see page 120)	Prepares all queries in the dataset to be executed.
	UnPrepare (see page 121)	Resets the state of a dataset's internal queries.

4.1.4.2.1 TIBDataSet.BatchInput Method

Executes the parameterized query in SQL for input in the referenced input object.

Pascal

```
procedure BatchInput(InputObject: TIBBatchInput);
```

C++

```
__fastcall BatchInput(TIBBatchInput InputObject);
```

Description

Call BatchInput to execute the parameterized query in SQL for input in the referenced input object.

4.1.4.2.2 TIBDataSet.BatchOutput Method

Outputs the selected query in SQL to the referenced OutputObject.

Pascal

```
procedure BatchOutput(OutputObject: TIBBatchOutput);
```

C++

```
__fastcall BatchOutput(TIBBatchOutput OutputObject);
```

Description

Call BatchOutput to output the selected query in SQL to the referenced OutputObject.

4.1.4.2.3 TIBDataSet.ExecSQL Method

Executes the SQL statement that the SelectSQL (see page 131) property specifies when it is not a SELECT statement.

Pascal

```
procedure ExecSQL;
```

C++

```
__fastcall ExecSQL();
```

Description

Call ExecSQL to execute the main query for this dataset when it is not a SELECT statement. If the value of the SelectSQL (see page 131) property is not set, or if it is set to a SELECT statement, ExecSQL raises an EIBError (see page 3) exception.

4.1.4.2.4 TIBDataSet.InternalOpen Method

Pascal

```
procedure InternalOpen; override;
```

C++

```
virtual __fastcall InternalOpen();
```

Description

This is InternalOpen, a member of class TIBDataSet.

4.1.4.2.5 TIBDataSet.ParamByName Method

Returns a specified parameter of the SELECT query that fetches this dataset's data.

Pascal

```
function ParamByName(Idx: String): TIBXSQLVAR;
```

C++

```
__fastcall TIBXSQLVAR ParamByName(UnicodeString Idx);
```

Description

Call ParamByName to retrieve the object that represents a parameter of the query that fetches this dataset's data. Use the TIBXSQLVAR (see page 544) object this method returns to get or set the parameter value.

Idx is the name of the parameter.

4.1.4.2.6 TIBDataSet.Prepare Method

Prepares all queries in the dataset to be executed.

Pascal

```
procedure Prepare;
```

C++

```
__fastcall Prepare();
```

Description

Call Prepare to prepare all queries in the dataset to be executed.

4.1.4.2.7 TIBDataSet.PSSetCommandText Method

Pascal

```
procedure PSSetCommandText(const CommandText: string); override;
```

C++

```
virtual __fastcall PSSetCommandText(const UnicodeString CommandText);
```

Description

This is PSSetCommandText, a member of class TIBDataSet.

4.1.4.2.8 TIBDataSet.SetFiltered Method

Pascal

```
procedure SetFiltered(Value: Boolean); override;
```

C++

```
virtual __fastcall SetFiltered(Boolean Value);
```

Description

This is SetFiltered, a member of class TIBDataSet.

4.1.4.2.9 TIBDataSet.UnPrepare Method

Resets the state of a dataset's internal queries.

Pascal

```
procedure UnPrepare;
```

C++

```
__fastcall UnPrepare();
```

Description

Note: Call UnPrepare to reset the state of a dataset's internal queries.

4.1.4.3 TIBDataSet Properties

The properties of the TIBDataSet class are listed here.

Public Properties

	Name	Description
	LiveMode (see page 128)	Indicates what types of edits, if any, can be made to the data in the dataset.
	Params (see page 130)	Provides the ability to specify values for a parameterized query.
R	Prepared (see page 131)	Determines whether or not a set of dataset queries is prepared for execution.
	SelectStmtHandle (see page 131)	Gets the handle for the SELECT statement.
	StatementType (see page 132)	Returns the statement type of the QSelect query.

Published Properties

	Name	Description
	Active (see page 122)	This is Active, a member of class TIBDataSet.
	AfterCancel (see page 122)	This is AfterCancel, a member of class TIBDataSet.
	AfterClose (see page 122)	This is AfterClose, a member of class TIBDataSet.
	AfterDelete (see page 123)	This is AfterDelete, a member of class TIBDataSet.
	AfterEdit (see page 123)	This is AfterEdit, a member of class TIBDataSet.
	AfterInsert (see page 123)	This is AfterInsert, a member of class TIBDataSet.
	AfterOpen (see page 123)	This is AfterOpen, a member of class TIBDataSet.
	AfterPost (see page 123)	This is AfterPost, a member of class TIBDataSet.
	AfterScroll (see page 124)	This is AfterScroll, a member of class TIBDataSet.
	AutoCalcFields (see page 124)	This is AutoCalcFields, a member of class TIBDataSet.
	BeforeCancel (see page 124)	This is BeforeCancel, a member of class TIBDataSet.
	BeforeClose (see page 124)	This is BeforeClose, a member of class TIBDataSet.
	BeforeDelete (see page 125)	This is BeforeDelete, a member of class TIBDataSet.
	BeforeEdit (see page 125)	This is BeforeEdit, a member of class TIBDataSet.
	BeforeInsert (see page 125)	This is BeforeInsert, a member of class TIBDataSet.
	BeforeOpen (see page 125)	This is BeforeOpen, a member of class TIBDataSet.
	BeforePost (see page 125)	This is BeforePost, a member of class TIBDataSet.
	BeforeScroll (see page 126)	This is BeforeScroll, a member of class TIBDataSet.
	BufferChunks (see page 126)	Specifies the dataset buffer chunk size.
	CachedUpdates (see page 126)	Specifies whether cached updates are enabled for a dataset.
	DataSource (see page 127)	This is DataSource, a member of class TIBDataSet.
	DeleteSQL (see page 127)	Holds the SQL statement used to delete rows from the dataset.
	Filtered (see page 127)	Specifies whether or not filtering is active for the dataset.
	GeneratorField (see page 127)	Specifies whether the dataset should refresh its data every time it posts a record.
	InsertSQL (see page 128)	Holds the SQL statement used to insert rows into the dataset.

	ModifySQL (see page 128)	Provides the ability to access the SQL object encapsulating the ModifySQL statement.
	OnCalcFields (see page 129)	This is OnCalcFields, a member of class TIBDataSet.
	OnDeleteError (see page 129)	This is OnDeleteError, a member of class TIBDataSet.
	OnEditError (see page 129)	This is OnEditError, a member of class TIBDataSet.
	OnFilterRecord (see page 129)	This is OnFilterRecord, a member of class TIBDataSet.
	OnNewRecord (see page 129)	This is OnNewRecord, a member of class TIBDataSet.
	OnPostError (see page 130)	This is OnPostError, a member of class TIBDataSet.
	ParamCheck (see page 130)	Specifies whether the parameter list for a query is regenerated if the SQL property changes at runtime.
	RefreshSQL (see page 131)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.
	SelectSQL (see page 131)	Provides the ability to directly access the SQL object encapsulating the SelectSQL statement.
	UniDirectional (see page 132)	Determines whether bidirectional cursors are enabled for the dataset.
	UpdateObject (see page 132)	Specifies the update object component used to update a read-only result set.

4.1.4.3.1 TIBDataSet.Active Property

Pascal

```
property Active;
```

C++

```
__property Active;
```

Description

This is Active, a member of class TIBDataSet.

4.1.4.3.2 TIBDataSet.AfterCancel Property

Pascal

```
property AfterCancel;
```

C++

```
__property AfterCancel;
```

Description

This is AfterCancel, a member of class TIBDataSet.

4.1.4.3.3 TIBDataSet.AfterClose Property

Pascal

```
property AfterClose;
```

C++

```
__property AfterClose;
```

Description

This is AfterClose, a member of class TIBDataSet.

4.1.4.3.4 TIBDataSet.AfterDelete Property

Pascal

```
property AfterDelete;
```

C++

```
__property AfterDelete;
```

Description

This is AfterDelete, a member of class TIBDataSet.

4.1.4.3.5 TIBDataSet.AfterEdit Property

Pascal

```
property AfterEdit;
```

C++

```
__property AfterEdit;
```

Description

This is AfterEdit, a member of class TIBDataSet.

4.1.4.3.6 TIBDataSet.AfterInsert Property

Pascal

```
property AfterInsert;
```

C++

```
__property AfterInsert;
```

Description

This is AfterInsert, a member of class TIBDataSet.

4.1.4.3.7 TIBDataSet.AfterOpen Property

Pascal

```
property AfterOpen;
```

C++

```
__property AfterOpen;
```

Description

This is AfterOpen, a member of class TIBDataSet.

4.1.4.3.8 TIBDataSet.AfterPost Property

Pascal

```
property AfterPost;
```

C++

```
__property AfterPost;
```

Description

This is AfterPost, a member of class TIBDataSet.

4.1.4.3.9 TIBDataSet.AfterScroll Property

Pascal

```
property AfterScroll;
```

C++

```
__property AfterScroll;
```

Description

This is AfterScroll, a member of class TIBDataSet.

4.1.4.3.10 TIBDataSet.AutoCalcFields Property

Pascal

```
property AutoCalcFields;
```

C++

```
__property AutoCalcFields;
```

Description

This is AutoCalcFields, a member of class TIBDataSet.

4.1.4.3.11 TIBDataSet.BeforeCancel Property

Pascal

```
property BeforeCancel;
```

C++

```
__property BeforeCancel;
```

Description

This is BeforeCancel, a member of class TIBDataSet.

4.1.4.3.12 TIBDataSet.BeforeClose Property

Pascal

```
property BeforeClose;
```

C++

```
__property BeforeClose;
```

Description

This is BeforeClose, a member of class TIBDataSet.

4.1.4.3.13 TIBDataSet.BeforeDelete Property

Pascal

```
property BeforeDelete;
```

C++

```
__property BeforeDelete;
```

Description

This is BeforeDelete, a member of class TIBDataSet.

4.1.4.3.14 TIBDataSet.BeforeEdit Property

Pascal

```
property BeforeEdit;
```

C++

```
__property BeforeEdit;
```

Description

This is BeforeEdit, a member of class TIBDataSet.

4.1.4.3.15 TIBDataSet.BeforeInsert Property

Pascal

```
property BeforeInsert;
```

C++

```
__property BeforeInsert;
```

Description

This is BeforeInsert, a member of class TIBDataSet.

4.1.4.3.16 TIBDataSet.BeforeOpen Property

Pascal

```
property BeforeOpen;
```

C++

```
__property BeforeOpen;
```

Description

This is BeforeOpen, a member of class TIBDataSet.

4.1.4.3.17 TIBDataSet.BeforePost Property

Pascal

```
property BeforePost;
```

C++

```
__property BeforePost;
```

Description

This is BeforePost, a member of class TIBDataSet.

4.1.4.3.18 TIBDataSet.BeforeScroll Property

Pascal

```
property BeforeScroll;
```

C++

```
__property BeforeScroll;
```

Description

This is BeforeScroll, a member of class TIBDataSet.

4.1.4.3.19 TIBDataSet.BufferChunks Property

Specifies the dataset buffer chunk size.

Pascal

```
property BufferChunks: Integer;
```

C++

```
__property int BufferChunks;
```

Description

Use BufferChunks to get or set the number of records for which the dataset allocates buffer space at any time. When the dataset's buffer is full, trying to fetch an additional record causes the dataset to reallocate the buffer so that it has enough memory to hold an additional BufferChunks records.

Note: When UniDirectional is true, BufferChunks has no meaning.

4.1.4.3.20 TIBDataSet.CachedUpdates Property

Specifies whether cached updates are enabled for a dataset.

Pascal

```
property CachedUpdates: Boolean;
```

C++

```
__property Boolean CachedUpdates;
```

Description

CachedUpdates enables or disables the use of cached updates for a dataset. If CachedUpdates is true, cached updates are enabled. If CachedUpdates is false, cached updates are disabled.

When cached updates are enabled, updates to a dataset (such as posting changes, inserting new records, or deleting records), are stored in an internal cache on the client machine instead of being written directly to the dataset's underlying database tables. When changes are complete, an application writes all cached changes to the database in the context of a single transaction.

Note: **Note**: Instead of using cached updates, applications can obtain the same benefits with greater control by using a client dataset.

4.1.4.3.21 TIBDataSet.DataSource Property

Pascal

```
property DataSource;
```

C++

```
__property DataSource;
```

Description

This is DataSource, a member of class TIBDataSet.

4.1.4.3.22 TIBDataSet.DeleteSQL Property

Holds the SQL statement used to delete rows from the dataset.

Pascal

```
property DeleteSQL: TStrings;
```

C++

```
__property TStrings DeleteSQL;
```

Description

Use DeleteSQL to delete rows in the dataset.

4.1.4.3.23 TIBDataSet.Filtered Property

Specifies whether or not filtering is active for the dataset.

Pascal

```
property Filtered;
```

C++

```
__property Filtered;
```

Description

Check Filtered to determine whether or not dataset filtering is in effect. If Filtered is true, then filtering is active. Otherwise Filtered is false. To apply filter conditions specified in the Filter property or the OnFilterRecord (see page 129) event handler, set Filtered to true.

Note: When filtering is enabled, user edits to a record may mean that the record no longer meets a filter's test condition. The next time the record is retrieved from the dataset while the filter is in effect, the record may seem to disappear. If that happens, the next record that passes the filter condition becomes the current record.

4.1.4.3.24 TIBDataSet.GeneratorField Property

Specifies whether the dataset should refresh its data every time it posts a record.

Pascal

```
property GeneratorField: TIBGeneratorField;
```

C++

```
__property TIBGeneratorField GeneratorField;
```

Description

Use GeneratorField to specify a generator that automatically provides the value for one of the fields in the dataset. Use the properties of the TIBGeneratorField object to indicate which field's value should be set and how that value should be calculated.

Warning: Do not set the value of GeneratorField. To change the value of this property, change the properties of the TIBGeneratorField object that is the value of this property.

4.1.4.3.25 TIBDataSet.InsertSQL Property

Holds the SQL statement used to insert rows into the dataset.

Pascal

```
property InsertSQL: TStrings;
```

C++

```
__property TStrings InsertSQL;
```

Description

Use InsertSQL to insert rows into the dataset.

4.1.4.3.26 TIBDataSet.LiveMode Property

Indicates what types of edits, if any, can be made to the data in the dataset.

Pascal

```
property LiveMode: TLiveModes;
```

C++

```
__property TLiveModes LiveMode;
```

Description

Read LiveMode to determine what types of edits should be allowed. LiveMode is set automatically to restrict the capabilities of the dataset to reflect the user's rights.

Note: For read-only datasets, such as TIBQuery, LiveMode is always an empty set.

4.1.4.3.27 TIBDataSet.ModifySQL Property

Provides the ability to access the SQL object encapsulating the ModifySQL statement.

Pascal

```
property ModifySQL: TStrings;
```

C++

```
__property TStrings ModifySQL;
```

Description

Use ModifySQL to access the SQL object that encapsulates the ModifySQL statement.

4.1.4.3.28 TIBDataSet.OnCalcFields Property

Pascal

```
property OnCalcFields;
```

C++

```
__property OnCalcFields;
```

Description

This is OnCalcFields, a member of class TIBDataSet.

4.1.4.3.29 TIBDataSet.OnDeleteError Property

Pascal

```
property OnDeleteError;
```

C++

```
__property OnDeleteError;
```

Description

This is OnDeleteError, a member of class TIBDataSet.

4.1.4.3.30 TIBDataSet.OnEditError Property

Pascal

```
property OnEditError;
```

C++

```
__property OnEditError;
```

Description

This is OnEditError, a member of class TIBDataSet.

4.1.4.3.31 TIBDataSet.OnFilterRecord Property

Pascal

```
property OnFilterRecord;
```

C++

```
__property OnFilterRecord;
```

Description

This is OnFilterRecord, a member of class TIBDataSet.

4.1.4.3.32 TIBDataSet.OnNewRecord Property

Pascal

```
property OnNewRecord;
```

C++

```
__property OnNewRecord;
```

Description

This is OnNewRecord, a member of class TIBDataSet.

4.1.4.3.33 TIBDataSet.OnPostError Property

Pascal

```
property OnPostError;
```

C++

```
__property OnPostError;
```

Description

This is OnPostError, a member of class TIBDataSet.

4.1.4.3.34 TIBDataSet.ParamCheck Property

Specifies whether the parameter list for a query is regenerated if the SQL property changes at runtime.

Pascal

```
property ParamCheck: Boolean;
```

C++

```
__property Boolean ParamCheck;
```

Description

Set ParamCheck to specify whether the Params property is cleared and regenerated if an application modifies the SQL in this dataset's query. By default ParamCheck is true, meaning that the Params property is automatically regenerated at runtime. When ParamCheck is true, the proper number of parameters is guaranteed to be generated for the current SQL statement.

This property is useful for data definition language (DDL) statements that contain parameters as part of the DDL statement that are not parameters for the TIBCustomDataSet. For example, the DDL statement to create a stored procedure may contain parameter statements that are part of the stored procedure. Set ParamCheck to false to prevent these parameters from being mistaken for parameters of the TIBCustomDataSet executing the DDL statement.

4.1.4.3.35 TIBDataSet.Params Property

Provides the ability to specify values for a parameterized query.

Pascal

```
property Params: TIBXSQLDA;
```

C++

```
__property TIBXSQLDA Params;
```

Description

Use Params to specify values for a parameterized query.

For example:

```
DataSet1.Params[0].AsInteger := 24
 DataSet1.Params.ByName['Field2'].AsString := 'foo'
    DataSet1->Params->Items[0]->AsInteger = 24;
 DataSet1->Params->ByName["Field2"]->AsString = "foo";
```

4.1.4.3.36 TIBDataSet.Prepared Property

Determines whether or not a set of dataset queries is prepared for execution.

Pascal

```
property Prepared: Boolean;
```

C++

```
__property Boolean Prepared;
```

Description

Examine Prepared to determine if a set of queries is already prepared for execution. If Prepared is true, the set of queries is prepared, and if Prepared is false, the set of queries is not prepared. While a set of queries need not be prepared before execution, execution performance is enhanced if the set of queries is prepared beforehand, particularly if it is a parameterized set of queries that is executed more than once using the same parameter values.

4.1.4.3.37 TIBDataSet.RefreshSQL Property

Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.

Pascal

```
property RefreshSQL: TStrings;
```

C++

```
__property TStrings RefreshSQL;
```

Description

Use RefreshSQL to access the SQL object that encapsulates the RefreshSQL statement.

4.1.4.3.38 TIBDataSet.SelectSQL Property

Provides the ability to directly access the SQL object encapsulating the SelectSQL statement.

Pascal

```
property SelectSQL: TStrings;
```

C++

```
__property TStrings SelectSQL;
```

Description

Use SelectSQL to access the SQL object that encapsulates the SelectSQL statement.

4.1.4.3.39 TIBDataSet.SelectStmtHandle Property

Gets the handle for the SELECT statement.

Pascal

```
property SelectStmtHandle: TISC_STMT_HANDLE;
```

C++

```
__property TISC_STMT_HANDLE SelectStmtHandle;
```

Description

Use SelectStmtHandle to get the handle for the SELECT statement.

4.1.4.3.40 TIBDataSet.StatementType Property

Returns the statement type of the QSelect query.

Pascal

```
property StatementType: TIBSQLTypes;
```

C++

```
__property TIBSQLTypes StatementType;
```

Description

Use StatementType to determine the statement type of a QSelect query.

4.1.4.3.41 TIBDataSet.UniDirectional Property

Determines whether bidirectional cursors are enabled for the dataset.

Pascal

```
property UniDirectional: Boolean;
```

C++

```
__property Boolean UniDirectional;
```

Description

Use UniDirectional to determine whether bidirectional cursors are enabled for this dataset.

Note: Do not confuse the UniDirectional property with the IsUniDirectional property inherited from TDataSet. IsUniDirectional indicates whether the dataset is a dbExpress dataset, which, in addition to using a unidirectional cursor, does not buffer records. UniDirectional is only concerned with whether the cursor permits backward or random navigation.

4.1.4.3.42 TIBDataSet.UpdateObject Property

Specifies the update object component used to update a read-only result set.

Pascal

```
property UpdateObject: TIBDataSetUpdateObject;
```

C++

```
__property TIBDataSetUpdateObject UpdateObject;
```

Description

Use UpdateObject to specify the TIBDataSetUpdateObject component to use in an application that must be able to update a read-only result set.

In some cases, such as when using TIBQuery, the dataset does not automatically update records. In these cases, UpdateObject can be used to specify a TIBUpdateSQL component that performs updates as a separate transaction that is transparent to the application.

TIBDataSetUpdateObject is especially useful for complicated updates from an OnUpdateRecord event.

4.1.4.4 TIBDataSet Events

The events of the TIBDataSet class are listed here.

Published Events

	Name	Description
	AfterDatabaseDisconnect (see page 133)	Occurs immediately after the connection to the associated database is closed.
	AfterTransactionEnd (see page 133)	Occurs immediately after a transaction that includes this dataset's query ends.
	BeforeDatabaseDisconnect (see page 133)	Occurs immediately before the connection to the associated database is closed.
	BeforeTransactionEnd (see page 134)	Occurs immediately before a transaction that includes this dataset's query ends.
	DatabaseFree (see page 134)	Occurs when the database associated with this dataset is freed.
	TransactionFree (see page 134)	Occurs when the transaction that includes this dataset's query is freed.

4.1.4.4.1 TIBDataSet.AfterDatabaseDisconnect Event

Occurs immediately after the connection to the associated database is closed.

Pascal

```
property AfterDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterDatabaseDisconnect;
```

Description

Write an AfterDatabaseDisconnect event handler to respond when the dataset closes its connection to the database specified by the Database property.

4.1.4.4.2 TIBDataSet.AfterTransactionEnd Event

Occurs immediately after a transaction that includes this dataset's query ends.

Pascal

```
property AfterTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterTransactionEnd;
```

Description

Write an AfterTransactionEnd event handler to respond when the transaction specified by the Transaction property ends.

4.1.4.4.3 TIBDataSet.BeforeDatabaseDisconnect Event

Occurs immediately before the connection to the associated database is closed.

Pascal

```
property BeforeDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeDatabaseDisconnect;
```

Description

Write a BeforeDatabaseDisconnect event handler to respond when the dataset is about to close its connection to the database

specified by the Database property.

4.1.4.4.4 TIBDataSet.BeforeTransactionEnd Event

Occurs immediately before a transaction that includes this dataset's query ends.

Pascal

```
property BeforeTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeTransactionEnd;
```

Description

Write a BeforeTransactionEnd event handler to respond when the transaction specified by the Transaction property is about to end.

4.1.4.4.5 TIBDataSet.DatabaseFree Event

Occurs when the database associated with this dataset is freed.

Pascal

```
property DatabaseFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent DatabaseFree;
```

Description

Write a DatabaseFree event handler to respond when the database specified by the Database property is freed.

4.1.4.4.6 TIBDataSet.TransactionFree Event

Occurs when the transaction that includes this dataset's query is freed.

Pascal

```
property TransactionFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent TransactionFree;
```

Description

Write a TransactionFree event handler to respond when the transaction specified by the Transaction property is freed.

4.1.5 TIBDataSetUpdateObject Class

TIBDataSetUpdateObject is the abstract base class for update objects used to update otherwise unupdateable queries when cached updates are enabled.

Class Hierarchy

Pascal

```
TIBDataSetUpdateObject = class(TComponent);
```

C++

```
class TIBDataSetUpdateObject : public TComponent;
```

File

IBCustomDataSet

Description

Use TIBDataSetUpdateObject as a base class when creating customized update objects that can be used to update datasets. TIBDataSetUpdateObject declares a single property and some abstract methods, but provides no implementation details: these must be provided by descendant objects.

4.1.5.1 TIBDataSetUpdateObject Members

The following tables list the members exposed by TIBDataSetUpdateObject.

Protected Methods

	Name	Description
	Apply (see page 136)	This is Apply, a member of class TIBDataSetUpdateObject.
	GetDataSet (see page 136)	This is GetDataSet, a member of class TIBDataSetUpdateObject.
	GetSQL (see page 137)	This is GetSQL, a member of class TIBDataSetUpdateObject.
	SetDataSet (see page 137)	This is SetDataSet, a member of class TIBDataSetUpdateObject.

Protected Properties

	Name	Description
	DataSet (see page 137)	This is DataSet, a member of class TIBDataSetUpdateObject.

Public Methods

	Name	Description
	Create (see page 135)	Creates an instance of a TIBDataSetUpdateObject (see page 134) component.
	Destroy (see page 136)	Frees all resources associated with this instance.

Published Properties

	Name	Description
	RefreshSQL (see page 137)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.

4.1.5.2 TIBDataSetUpdateObject.Create Constructor

Creates an instance of a TIBDataSetUpdateObject (see page 134) component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBDataSetUpdateObject(TComponent * AOwner);
```

Description

Call Create to create an instance of a TIBDataSetUpdateObject (see page 134) component.

4.1.5.3 TIBDataSetUpdateObject.Destroy Destructor

Frees all resources associated with this instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBDataSetUpdateObject();
```

Description

Do not call Destroy directly in an application. Usually destruction of objects is handled automatically by Delphi. If an application creates its own instance of an update object, however, the application should call Free, which verifies that the update object is not already freed before calling Destroy.

4.1.5.4 TIBDataSetUpdateObject Methods

The methods of the TIBDataSetUpdateObject class are listed here.

Protected Methods

	Name	Description
	Apply (see page 136)	This is Apply, a member of class TIBDataSetUpdateObject.
	GetDataSet (see page 136)	This is GetDataSet, a member of class TIBDataSetUpdateObject.
	GetSQL (see page 137)	This is GetSQL, a member of class TIBDataSetUpdateObject.
	SetDataSet (see page 137)	This is SetDataSet, a member of class TIBDataSetUpdateObject.

4.1.5.4.1 TIBDataSetUpdateObject.Apply Method

Pascal

```
procedure Apply(UpdateKind: TUpdateKind); virtual; abstract;
```

C++

```
virtual __fastcall Apply(TUpdateKind UpdateKind) = 0;
```

Description

This is Apply, a member of class TIBDataSetUpdateObject.

4.1.5.4.2 TIBDataSetUpdateObject.GetDataSet Method

Pascal

```
function GetDataSet: TIBCustomDataSet; virtual; abstract;
```

C++

```
virtual __fastcall TIBCustomDataSet GetDataSet() = 0;
```

Description

This is GetDataSet, a member of class TIBDataSetUpdateObject.

4.1.5.4.3 TIBDataSetUpdateObject.GetSQL Method

Pascal

```
function GetSQL(UpdateKind: TUpdateKind): TStrings; virtual; abstract;
```

C++

```
virtual __fastcall TStrings GetSQL(TUpdateKind UpdateKind) = 0;
```

Description

This is GetSQL, a member of class TIBDataSetUpdateObject.

4.1.5.4.4 TIBDataSetUpdateObject.SetDataSet Method

Pascal

```
procedure SetDataSet(ADataSet: TIBCustomDataSet); virtual; abstract;
```

C++

```
virtual __fastcall SetDataSet(TIBCustomDataSet ADataSet) = 0;
```

Description

This is SetDataSet, a member of class TIBDataSetUpdateObject.

4.1.5.5 TIBDataSetUpdateObject Properties

The properties of the TIBDataSetUpdateObject class are listed here.

Protected Properties

	Name	Description
	DataSet (see page 137)	This is DataSet, a member of class TIBDataSetUpdateObject.

Published Properties

	Name	Description
	RefreshSQL (see page 137)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.

4.1.5.5.1 TIBDataSetUpdateObject.DataSet Property

Pascal

```
property DataSet: TIBCustomDataSet;
```

C++

```
__property TIBCustomDataSet DataSet;
```

Description

This is DataSet, a member of class TIBDataSetUpdateObject.

4.1.5.5.2 TIBDataSetUpdateObject.RefreshSQL Property

Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.

Pascal

```
property RefreshSQL: TStrings;
```

C++

```
__property TStrings RefreshSQL;
```

Description

Use RefreshSQL to access the SQL object that encapsulates the RefreshSQL statement.

4.1.6 TIBDSBlobStream Class

TIBDSBlobStream reads from and writes to BLOB fields on behalf of an IB (see page 1) Dataset.

Class Hierarchy

Pascal

```
TIBDSBlobStream = class(TStream);
```

C++

```
class TIBDSBlobStream : public TStream;
```

File

IBCustomDataSet

Description

TIBDSBlobStream is the type of stream returned when an application calls an IB (see page 1) dataset's CreateBlobStream method. Use this stream to access or modify the value of a Blob field in the dataset. TIBDSBlobStream allows objects that have no specialized knowledge of how data is stored in a Blob field to read or write such data by employing the uniform stream interface.

To use TIBDSBlobStream, call CreateBlobStream to obtain an instance of the stream and use the methods of the stream to read or write the data. The dataset retains ownership of the stream, so you should not free this Blob stream when you are done.

4.1.6.1 TIBDSBlobStream Members

The following tables list the members exposed by TIBDSBlobStream.

Protected Fields

	Name	Description
	FBlobStream (see page 139)	This is FBlobStream, a member of class TIBDSBlobStream.
	FField (see page 140)	This is FField, a member of class TIBDSBlobStream.
	FModified (see page 140)	This is FModified, a member of class TIBDSBlobStream.

Public Methods

	Name	Description
	Create (see page 139)	Resets the stream to the beginning of the Blob.
V	Destroy (see page 139)	Frees the instance of TIBDSBlobStream (see page 138).
V	Read (see page 140)	Reads the requested number of bytes from the Blob.
V	Seek (see page 141)	Resets the current position of the TIBDSBlobStream (see page 138) object.

	SetSize (see page 141)	Set the size of the Blob to the requested size.
	Write (see page 141)	Sets the field to be modified.

4.1.6.2 TIBDSBlobStream.Create Constructor

Resets the stream to the beginning of the Blob.

Pascal

```
constructor Create(AField: TField; ABlobStream: TIBBlobStream; Mode: TBlobStreamMode);
```

C++

```
__fastcall TIBDSBlobStream(TField AField, TIBBlobStream ABlobStream, TBlobStreamMode Mode);
```

Description

Call Create to reset the stream to the beginning of the Blob. Depending on the mode, it also truncates the Blob stream.

4.1.6.3 TIBDSBlobStream.Destroy Destructor

Frees the instance of TIBDSBlobStream (see page 138).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBDSBlobStream();
```

Description

Destroy frees the blob stream object and writes the BLOB value to the underlying TBlobField object.

Do not explicitly destroy the instance of TIBDSBlobStream (see page 138) that you obtain by calling a dataset's CreateBlobStream method. The Dataset retains ownership of this stream and is responsible for freeing it.

When working with a Blob stream that you explicitly create and destroy, use the TIBBlobStream (see page 40) class instead.

4.1.6.4 TIBDSBlobStream Fields

The fields of the TIBDSBlobStream class are listed here.

Protected Fields

	Name	Description
	FBlobStream (see page 139)	This is FBlobStream, a member of class TIBDSBlobStream.
	FField (see page 140)	This is FField, a member of class TIBDSBlobStream.
	FModified (see page 140)	This is FModified, a member of class TIBDSBlobStream.

4.1.6.4.1 TIBDSBlobStream.FBlobStream Field

Pascal

```
FBlobStream: TIBBlobStream;
```

C++

```
TIBBlobStream FBlobStream;
```

Description

This is FBlobStream, a member of class TIBDSBlobStream.

4.1.6.4.2 TIBDSBlobStream.FField Field

Pascal

```
FField: TField;
```

C++

```
TField FField;
```

Description

This is FField, a member of class TIBDSBlobStream.

4.1.6.4.3 TIBDSBlobStream.FModified Field

Pascal

```
FModified: Boolean;
```

C++

```
Boolean FModified;
```

Description

This is FModified, a member of class TIBDSBlobStream.

4.1.6.5 TIBDSBlobStream Methods

The methods of the TIBDSBlobStream class are listed here.

Public Methods

	Name	Description
	Read (see page 140)	Reads the requested number of bytes from the Blob.
	Seek (see page 141)	Resets the current position of the TIBDSBlobStream (see page 138) object.
	SetSize (see page 141)	Set the size of the Blob to the requested size.
	Write (see page 141)	Sets the field to be modified.

4.1.6.5.1 TIBDSBlobStream.Read Method

Reads the requested number of bytes from the Blob.

Pascal

```
function Read(var Buffer; Count: Longint): Longint; override;
```

C++

```
virtual __fastcall Longint Read( Buffer, Longint Count);
```

Description

Call Read to read data from the Blob field when the number of bytes in the field's data is not known. Buffer must have at least Count bytes allocated to hold the data that was read from the field.

Read transfers up to Count bytes from the Blob data into Buffer, starting in the current position, and then advances the current

position by the number of bytes actually transferred. Read returns the number of bytes actually transferred (which may be less than the number requested in Count.)

Read checks the Transliterate property of the field, and converts the data into ANSI from the character set specified by the dataset if Transliterate is true.

All the other data-reading methods of a Blob stream (ReadBuffer, ReadComponent) call Read to do their actual reading.

Do not call Read when the TIBDSBlobStream (see page 138) was created in bmWrite mode.

4.1.6.5.2 TIBDSBlobStream.Seek Method

Resets the current position of the TIBDSBlobStream (see page 138) object.

Pascal

```
function Seek(Offset: Longint; Origin: Word): Longint; override;
```

C++

```
virtual __fastcall Longint Seek(Longint Offset, Word Origin);
```

Description

Use Seek to move the current position within the Blob data by the indicated offset. Seek allows an application to read from or write to a particular location within the Blob data.

The Origin parameter indicates how to interpret the Offset parameter. Origin should be one of the following values:

Value	Meaning
soFromBeginning	Offset is from the beginning of the Blob data Seek moves to the position Offset Offset must be >= 0
soFromCurrent	Offset is from the current position in the Blob data Seek moves to Position + Offset
soFromEnd	Offset is from the end of the Blob data Offset must be <= 0 to indicate a number of bytes before the end of the Blob

Seek returns the new value of the Position property, the new current position in the Blob data.

4.1.6.5.3 TIBDSBlobStream.SetSize Method

Set the size of the Blob to the requested size.

Pascal

```
procedure SetSize(NewSize: Longint); override;
```

C++

```
virtual __fastcall SetSize(Longint NewSize);
```

Description

Call SetSize to set the size of the Blob to the requested size.

4.1.6.5.4 TIBDSBlobStream.Write Method

Sets the field to be modified.

Pascal

```
function Write(const Buffer; Count: Longint): Longint; override;
```

C++

```
virtual __fastcall Longint Write(const  Buffer, Longint Count);
```

Description

Use Write to set the field to be modified, write the requested number of bytes to the Blob stream.

4.1.7 TIBGeneratorField Class

TIBGeneratorField calls the server to generate values for a field in an IB (see page 1) dataset.

Class Hierarchy

Pascal

```
TIBGeneratorField = class(TPersistent);
```

C++

```
class TIBGeneratorField : public TPersistent;
```

File

IBCustomDataSet

Description

Assign (see page 144) an instance of TIBGeneratorField as the value of a dataset's GeneratorField property to indicate that the value of a specified field in the dataset can be generated by the server.

Use the Field (see page 145) property to specify the field whose value is generated by the TIBGeneratorField object. Use the Generator (see page 145) and IncrementBy (see page 146) properties to specify how the server should generate field values. Use the ApplyEvent (see page 145) property to indicate when the TIBGeneratorField should call the server to generate field values.

4.1.7.1 TIBGeneratorField Members

The following tables list the members exposed by TIBGeneratorField.

Public Methods

	Name	Description
	Apply (see page 143)	Calls the generator on the server to generate a field value and assigns it to the associated field.
	Assign (see page 144)	Copies the properties of another object.
	Create (see page 143)	Creates an instance of TIBGeneratorField (see page 142).
	ValueName (see page 144)	Returns a string that describes the value that the IB (see page 1) generate fetches.

Published Properties

	Name	Description
	ApplyEvent (see page 145)	Indicates when the TIBGeneratorField (see page 142) instance calls the server to generate a value for the associated field.

	Field (see page 145)	Specifies the field whose value the TIBGeneratorField (see page 142) object generates.
	Generator (see page 145)	Names the generator on the server that TIBGeneratorField (see page 142) calls to generate field values.
	IncrementBy (see page 146)	Names the amount by which a generator on the server increments the value of the associated field.

4.1.7.2 TIBGeneratorField.Create Constructor

Creates an instance of TIBGeneratorField (see page 142).

Pascal

```
constructor Create(ADataSet: TIBCustomDataSet);
```

C++

```
__fastcall TIBGeneratorField(TIBCustomDataSet ADataSet);
```

Description

Do not create instances of TIBGeneratorField (see page 142). This class is designed to work with an IB (see page 1) dataset and is automatically created by the dataset to implement its GeneratorField property.

ADataSet is the dataset that creates the TIBGeneratorField (see page 142) instance and which contains the field whose value the generator field assigns.

4.1.7.3 TIBGeneratorField Methods

The methods of the TIBGeneratorField class are listed here.

Public Methods

	Name	Description
	Apply (see page 143)	Calls the generator on the server to generate a field value and assigns it to the associated field.
	Assign (see page 144)	Copies the properties of another object.
	ValueName (see page 144)	Returns a string that describes the value that the IB (see page 1) generate fetches.

4.1.7.3.1 TIBGeneratorField.Apply Method

Calls the generator on the server to generate a field value and assigns it to the associated field.

Pascal

```
procedure Apply;
```

C++

```
__fastcall Apply();
```

Description

Call Apply to fetch the server-generated value for the field identified by the Field (see page 145) property and assign that value to the field.

If the ApplyEvent (see page 145) property is not gamOnServer, the associated dataset calls Apply automatically at the point indicated by ApplyEvent (see page 145).

4.1.7.3.2 TIBGeneratorField.Assign Method

Copies the properties of another object.

Pascal

```
procedure Assign(Source: TPersistent); override;
```

C++

```
virtual __fastcall Assign(TPersistent * Source);
```

Description

Call Assign to copy the properties of another object on to this TIBGeneratorField (see page 142) instance. TIBGeneratorField (see page 142) overrides the inherited method to copy the Field (see page 145), Generator (see page 145), IncrementBy (see page 146) and ApplyEvent (see page 145) properties of the object specified by Source if it is another TIBGeneratorField (see page 142) instance.

If Source is not another TIBGeneratorField (see page 142), Assign calls the inherited method, which allows any objects that know how to assign themselves to a TIBGeneratorField (see page 142) to copy their properties.

4.1.7.3.3 TIBGeneratorField.ValueName Method

Returns a string that describes the value that the IB (see page 1) generate fetches.

Pascal

```
function ValueName: string;
```

C++

```
__fastcall UnicodeString ValueName();
```

Description

ValueName returns a string of the form

```
Generator -> Field By IncrementBy
```

where Generator (see page 145) is the value of the Generator (see page 145) property, Field (see page 145) is the value of the Field (see page 145) property, and IncrementBy (see page 146) is the value of the IncrementBy (see page 146) property.

4.1.7.4 TIBGeneratorField Properties

The properties of the TIBGeneratorField class are listed here.

Published Properties

	Name	Description
	ApplyEvent (see page 145)	Indicates when the TIBGeneratorField (see page 142) instance calls the server to generate a value for the associated field.
	Field (see page 145)	Specifies the field whose value the TIBGeneratorField (see page 142) object generates.
	Generator (see page 145)	Names the generator on the server that TIBGeneratorField (see page 142) calls to generate field values.
	IncrementBy (see page 146)	Names the amount by which a generator on the server increments the value of the associated field.

4.1.7.4.1 TIBGeneratorField.ApplyEvent Property

Indicates when the TIBGeneratorField (see page 142) instance calls the server to generate a value for the associated field.

Pascal

```
property ApplyEvent: TIBGeneratorApplyEvent;
```

C++

```
__property TIBGeneratorApplyEvent ApplyEvent;
```

Description

Set TIBGeneratorApplyEvent (see page 151) to indicate when the server generates field values using this generator.

Note: Regardless of the value of ApplyEvent, applications can generate the field value at any time by calling the Apply (see page 143) method.

4.1.7.4.2 TIBGeneratorField.Field Property

Specifies the field whose value the TIBGeneratorField (see page 142) object generates.

Pascal

```
property Field: string;
```

C++

```
__property UnicodeString Field;
```

Description

Set Field to the name of the field whose value the generator sets. When the event specified by the ApplyEvent (see page 145) property occurs, or when an application calls the Apply (see page 143) method, the TIBGeneratorField (see page 142) object calls the server to generate a field value and assigns that value to the specified field.

Field must be the FieldName property of one of the fields in the associated dataset's Fields property list. The TIBGeneratorField (see page 142) object is associated with a dataset when it is the value of that dataset's GeneratorField property.

4.1.7.4.3 TIBGeneratorField.Generator Property

Names the generator on the server that TIBGeneratorField (see page 142) calls to generate field values.

Pascal

```
property Generator: string;
```

C++

```
__property UnicodeString Generator;
```

Description

TIBGeneratorField (see page 142) uses Generator, along with the IncrementBy (see page 146) property, to generate the SQL statement that fetches the generated field value. The generated SQL statement has the form

SELECT GEN_ID(Generator, IncrementBy (see page 146)) FROM RDB$DATABASE

Where 'Generator' is replaced by the value of Generator and 'IncrementBy (see page 146)' is replaced by the value of IncrementBy (see page 146).

4.1.7.4.4 TIBGeneratorField.IncrementBy Property

Names the amount by which a generator on the server increments the value of the associated field.

Pascal

```
property IncrementBy: Integer;
```

C++

```
__property int IncrementBy;
```

Description

TIBGeneratorField (see page 142) uses IncrementBy, along with the Generator (see page 145) property, to generate the SQL statement that fetches the generated field value. The generated SQL statement has the form

SELECT GEN_ID(Generator (see page 145), IncrementBy) FROM RDB$DATABASE

Where 'Generator (see page 145)' is replaced by the value of Generator (see page 145) and 'IncrementBy' is replaced by the value of IncrementBy.

4.1.8 TIBStringField Class

TIBStringField allows for strings in excess of 8196 bytes of data.

Class Hierarchy

Pascal

```
TIBStringField = class(TStringField);
```

C++

```
class TIBStringField : public TStringField;
```

File

IBCustomDataSet

Description

A value of a string field is physically stored as a sequence of characters. Common uses for string fields are to store text, such as names and addresses.

TIBStringField introduces properties to translate between string values and other data types, and to manage language driver conversions. As a descendent of TStringField, TIBStringField inherits many properties, methods, and events that are useful for managing the value and properties of a field in a database. TIBStringField allows for strings in excess of 8196 bytes of data.

4.1.8.1 TIBStringField Members

The following tables list the members exposed by TIBStringField.

Public Methods

	Name	Description
	CheckTypeSize (see page 147)	Indicates the type size of the string field.
	Create (see page 147)	Creates an instance of a TIBStringField (see page 146) object.

	GetAsString (see page 148)	Returns the value of a field as a string.
	GetAsVariant (see page 148)	Returns the value of a field as a variant.
	GetValue (see page 148)	Returns the value of a field.
	SetAsString (see page 148)	Sets the value of a field as a string type.

4.1.8.2 TIBStringField.Create Constructor

Creates an instance of a TIBStringField (see page 146) object.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBStringField(TComponent * AOwner);
```

Description

It is seldom necessary to call Create directly, because a string field component is instantiated automatically for all string fields in a dataset.

After calling the inherited constructor, Create sets

- DataType to ftString.
- Size to 20.
- Transliterate to True.
- BlanksToNull to True

4.1.8.3 TIBStringField Methods

The methods of the TIBStringField class are listed here.

Public Methods

	Name	Description
	CheckTypeSize (see page 147)	Indicates the type size of the string field.
	GetAsString (see page 148)	Returns the value of a field as a string.
	GetAsVariant (see page 148)	Returns the value of a field as a variant.
	GetValue (see page 148)	Returns the value of a field.
	SetAsString (see page 148)	Sets the value of a field as a string type.

4.1.8.3.1 TIBStringField.CheckTypeSize Method

Indicates the type size of the string field.

Pascal

```
class procedure CheckTypeSize(Value: Integer); override;
```

C++

```
virtual __fastcall CheckTypeSize(int Value);
```

Description

Call CheckTypeSize to determine the type size of the string field.

There is both a virtual method and a static method for the CheckTypeSize method.

All sizes are valid for a TIBString field.

4.1.8.3.2 TIBStringField.GetAsString Method

Returns the value of a field as a string.

Pascal

```
function GetAsString: string; override;
```

C++

```
virtual __fastcall UnicodeString GetAsString();
```

Description

Call GetAsString to return the value of a field as a string.

4.1.8.3.3 TIBStringField.GetAsVariant Method

Returns the value of a field as a variant.

Pascal

```
function GetAsVariant: Variant; override;
```

C++

```
virtual __fastcall Variant GetAsVariant();
```

Description

Call GetAsVariant return the value of a field as type Variant.

4.1.8.3.4 TIBStringField.GetValue Method

Returns the value of a field.

Pascal

```
function GetValue(var Value: string): Boolean;
```

C++

```
__fastcall Boolean GetValue(UnicodeString Value);
```

Description

Call GetValue to return the value of a field.

4.1.8.3.5 TIBStringField.SetAsString Method

Sets the value of a field as a string type.

Pascal

```
procedure SetAsString(const Value: string); override;
```

C++

```
virtual __fastcall SetAsString(const UnicodeString Value);
```

Description

Call SetAsString to set the value of a field as a string type.

4.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TCachedUpdateStatus (see page 149)	TCachedUpdateStatus indicates the status of a record in a dataset that has cached updates.
	TIBGeneratorApplyEvent (see page 151)	TIBGeneratorApplyEvent indicates when a generator should create field values.
	TIBUpdateAction (see page 151)	TIBUpdateAction indicates the action that a dataset should take after an event handler has responded to an update error.
	TLiveMode (see page 152)	This is record IBCustomDataSet.TLiveMode.

Records

	Name	Description
	TFieldData (see page 150)	This is record IBCustomDataSet.TFieldData.
	TIBDBKey (see page 150)	TIBDBKey stores the key for a row in an InterBase table.
	TRecordData (see page 153)	This is record IBCustomDataSet.TRecordData.

4.2.1 IBCustomDataSet.TCachedUpdateStatus Enumeration

TCachedUpdateStatus indicates the status of a record in a dataset that has cached updates.

Pascal

```
TCachedUpdateStatus = (
  cusUnmodified,
  cusModified,
  cusInserted,
  cusDeleted,
  cusUninserted
);
```

C++

```
enum TCachedUpdateStatus {
  cusUnmodified,
  cusModified,
  cusInserted,
  cusDeleted,
  cusUninserted
};
```

File

IBCustomDataSet

Description

TCachedUpdateStatus values are:

Value Meaning

cusModified Modified records.

cusInserted Inserted records.

cusDeleted Deleted records.

cusUnmodified Unmodified records.

cusUninserted Uninserted records.

4.2.2 IBCustomDataSet.TFieldData Record

Pascal

```
TFieldData = record
  fdDataType: Short;
  fdDataScale: Short;
  fdNullable: Boolean;
  fdIsNull: Boolean;
  fdDataSize: Short;
  fdDataLength: Short;
  fdDataOfs: Integer;
end;
```

C++

```
struct TFieldData {
  Short fdDataType;
  Short fdDataScale;
  Boolean fdNullable;
  Boolean fdIsNull;
  Short fdDataSize;
  Short fdDataLength;
  int fdDataOfs;
};
```

File

IBCustomDataSet

Description

This is record IBCustomDataSet.TFieldData.

4.2.3 IBCustomDataSet.TIBDBKey Record

TIBDBKey stores the key for a row in an InterBase table.

Pascal

```
TIBDBKey = record
  DBKey: array[0..7] of Byte;
end;
```

C++

```
struct TIBDBKey {
  array[0..7] of Byte DBKey;
};
```

File

IBCustomDataSet

Description

TIBDBKey stores the unique row identifier for the current row in the current transaction.

4.2.4 IBCustomDataSet.TIBGeneratorApplyEvent Enumeration

TIBGeneratorApplyEvent indicates when a generator should create field values.

Pascal

```
TIBGeneratorApplyEvent = (
  gamOnNewRecord,
  gamOnPost,
  gamOnServer
);
```

C++

```
enum TIBGeneratorApplyEvent {
  gamOnNewRecord,
  gamOnPost,
  gamOnServer
};
```

File

IBCustomDataSet

Description

TIBGeneratorApplyEvent values are:

Value Meaning

gamOnNewRecord The dataset calls the server to generate the field value immediately after it inserts or appends a new record (before the OnNewRecord event).

gamOnPost The dataset calls the server to generate the field value immediately before it posts a record to the server (before the BeforePost event).

gamOnServer The dataset does not need to call the server to generate field values: the server generates these values automatically. With gamOnServer, TIBGeneratorField (see page 142) serves primarily to ensure that the user is not required to enter a value in the field.

See Also

Procedural types

Creating events

4.2.5 IBCustomDataSet.TIBUpdateAction Enumeration

TIBUpdateAction indicates the action that a dataset should take after an event handler has responded to an update error.

Pascal

```
TIBUpdateAction = (
```

```
    uaFail,
    uaAbort,
    uaSkip,
    uaRetry,
    uaApply,
    uaApplied
  );
```

C++

```
  enum TIBUpdateAction {
    uaFail,
    uaAbort,
    uaSkip,
    uaRetry,
    uaApply,
    uaApplied
  };
```

File

IBCustomDataSet

Description

TIBUpdateAction values are:

Value Meaning

uaFail Abort the update operation and display an error message

uaAbort Abort the update operation without displaying an error message

uaSkip Skip updating the record that raised the error condition, and leave the unapplied changes in the cache

uaRetry Repeat the update operation that originally raised the error condition

uaApplied The event handler applied the update (not used in error handling event handlers). Free the current record update from the cache.

uaApply For internal use only

4.2.6 IBCustomDataSet.TLiveMode Enumeration

Pascal

```
  TLiveMode = (
    lmInsert,
    lmModify,
    lmDelete,
    lmRefresh
  );
```

C++

```
  enum TLiveMode {
    lmInsert,
    lmModify,
    lmDelete,
    lmRefresh
  };
```

File

IBCustomDataSet

Description

This is record IBCustomDataSet.TLiveMode.

4.2.7 IBCustomDataSet.TRecordData Record

Pascal

```
TRecordData = record
  rdBookmarkFlag: TBookmarkFlag;
  rdFieldCount: Short;
  rdRecordNumber: Long;
  rdCachedUpdateStatus: TCachedUpdateStatus;
  rdUpdateStatus: TUpdateStatus;
  rdSavedOffset: DWORD;
  rdDBKey: TIBDBKey;
  rdFields: array[1..1] of TFieldData;
end;
```

C++

```
struct TRecordData {
  TBookmarkFlag rdBookmarkFlag;
  Short rdFieldCount;
  Long rdRecordNumber;
  TCachedUpdateStatus rdCachedUpdateStatus;
  TUpdateStatus rdUpdateStatus;
  DWORD rdSavedOffset;
  TIBDBKey rdDBKey;
  array[1..1] of TFieldData rdFields;
};
```

File

IBCustomDataSet

Description

This is record IBCustomDataSet.TRecordData.

4.3 Types

The following table lists types in this documentation.

Types

Name	Description
PBlobDataArray (see page 154)	This is type IBCustomDataSet.PBlobDataArray.
PDateTime (see page 154)	This is type IBCustomDataSet.PDateTime.
PFieldData (see page 155)	This is type IBCustomDataSet.PFieldData.
PIBDBKey (see page 155)	This is type IBCustomDataSet.PIBDBKey.
PRecordData (see page 155)	This is type IBCustomDataSet.PRecordData.
TBlobDataArray (see page 156)	This is type IBCustomDataSet.TBlobDataArray.
TIBUpdateErrorEvent (see page 156)	TIBUpdateErrorEvent is the type of event handlers that respond when an IBX dataset encounters an error applying a cached update.
TIBUpdateRecordEvent (see page 157)	TIBUpdateRecordEvent is the type of event handlers that respond when an IBX dataset applies a cached update.
TIBUpdateRecordTypes (see page 157)	TIBUpdateRecordTypes is a subset of all possible record types in a dataset that caches updates.
TLiveModes (see page 157)	TLiveModes and TLiveModes indicate what operations an application can perform on an IBX dataset.

4.3.1 IBCustomDataSet.PBlobDataArray Type

Pascal

```
PBlobDataArray = ^TBlobDataArray;
```

C++

```
^TBlobDataArray PBlobDataArray;
```

File

IBCustomDataSet

Description

This is type IBCustomDataSet.PBlobDataArray.

4.3.2 IBCustomDataSet.PDateTime Type

Pascal

```
PDateTime = ^TDateTime;
```

C++

```
^TDateTime PDateTime;
```

File

IBCustomDataSet

Description

This is type IBCustomDataSet.PDateTime.

4.3.3 IBCustomDataSet.PFieldData Type

Pascal

```
PFieldData = ^TFieldData;
```

C++

```
^TFieldData PFieldData;
```

File

IBCustomDataSet

Description

This is type IBCustomDataSet.PFieldData.

4.3.4 IBCustomDataSet.PIBDBKey Type

Pascal

```
PIBDBKey = ^TIBDBKey;
```

C++

```
^TIBDBKey PIBDBKey;
```

File

IBCustomDataSet

Description

This is type IBCustomDataSet.PIBDBKey.

4.3.5 IBCustomDataSet.PRecordData Type

Pascal

```
PRecordData = ^TRecordData;
```

C++

```
^TRecordData PRecordData;
```

File

IBCustomDataSet

Description

This is type IBCustomDataSet.PRecordData.

4.3.6 IBCustomDataSet.TBlobDataArray Type

Pascal

```
TBlobDataArray = array[0..0] of TIBBlobStream;
```

C++

```
array[0..0] of TIBBlobStream TBlobDataArray;
```

File

IBCustomDataSet

Description

This is type IBCustomDataSet.TBlobDataArray.

4.3.7 IBCustomDataSet.TIBUpdateErrorEvent Type

TIBUpdateErrorEvent is the type of event handlers that respond when an IBX dataset encounters an error applying a cached update.

Pascal

```
TIBUpdateErrorEvent = procedure (DataSet: TDataSet; E: EDatabaseError; UpdateKind:
TUpdateKind; var UpdateAction: TIBUpdateAction) of object;
```

C++

```
(DataSet: TDataSet; E: EDatabaseError; UpdateKind: TUpdateKind; var UpdateAction:
TIBUpdateAction) ( TIBUpdateErrorEvent)();
```

File

IBCustomDataSet

Description

DataSet is the TIBCustomDataSet (see page 62) descendant that is attempting to apply cached updates.

E is an EDBEngineError object from which an application can extract an error message and the actual cause of the error condition. An OnUpdateError handler can use this information to determine how to respond to the error condition.

UpdateKind indicates whether the error occurred while inserting, deleting, or modifying a record.

UpdateAction indicates the action to take when the OnUpdateError handler exits. On entry into the handler, UpdateAction is always set to uaFail. If OnUpdateError can handle or correct the error, set UpdateAction to uaRetry before exiting the error handler.

See Also

Procedural types

Creating events

4.3.8 IBCustomDataSet.TIBUpdateRecordEvent Type

TIBUpdateRecordEvent is the type of event handlers that respond when an IBX dataset applies a cached update.

Pascal

```
TIBUpdateRecordEvent = procedure (DataSet: TDataSet; UpdateKind: TUpdateKind; var
UpdateAction: TIBUpdateAction) of object;
```

C++

```
(DataSet: TDataSet; UpdateKind: TUpdateKind; var UpdateAction: TIBUpdateAction) (
TIBUpdateRecordEvent)();
```

File

IBCustomDataSet

Description

DataSet is the dataset that is applying updates.

UpdateKind whether the current update is the insertion of a record, the deletion of a record, or the modification of a record.

UpdateAction indicates the action taken by the OnUpdateRecord handler before it exits. On entry into the handler, UpdateAction is always set to uaFail. If OnUpdateRecord is successful, it should set UpdateAction to uaApplied before exiting.

See Also

Procedural types

Creating events

4.3.9 IBCustomDataSet.TIBUpdateRecordTypes Type

TIBUpdateRecordTypes is a subset of all possible record types in a dataset that caches updates.

Pascal

```
TIBUpdateRecordTypes = set of TCachedUpdateStatus;
```

C++

```
TCachedUpdateStatus TIBUpdateRecordTypes;
```

File

IBCustomDataSet

Description

TIBUpdateRecordTypes is a set of TCachedUpdateStatus (see page 149) values. It describes the different types of status allowed for records in a dataset that caches updates if they are to be included in a specified set.

4.3.10 IBCustomDataSet.TLiveModes Type

TLiveModes and TLiveModes indicate what operations an application can perform on an IBX dataset.

Pascal

```
TLiveModes = set of TLiveMode;
```

C++

```
TLiveMode TLiveModes;
```

File

IBCustomDataSet

Description

TLiveModes is a set of TLiveMode (see page 152) values. The following table lists the possible values:

Value	Meaning
lmInsert	The application can insert records.
lmModify	The application can modify existing records.
lmDelete	The application can delete records.
lmRefresh	The application can refresh the records in the dataset.

4.4 Variables

The following table lists variables in this documentation.

Variables

Name	Description
CreateProviderProc (see page 159)	This is variable IBCustomDataSet.CreateProviderProc.

4.4.1 IBCustomDataSet.CreateProviderProc Variable

Pascal

```
CreateProviderProc: function(DataSet: TIBCustomDataSet): IProviderSupport = nil;
```

C++

```
function(DataSet: TIBCustomDataSet): IProviderSupport CreateProviderProc = nil;
```

File

IBCustomDataSet

Description

This is variable IBCustomDataSet.CreateProviderProc.

4.5 Constants

The following table lists constants in this documentation.

Constants

Name	Description
BufferCacheSize (see page 160)	This is constant IBCustomDataSet.BufferCacheSize.
DefaultFieldClasses (see page 160)	This is constant IBCustomDataSet.DefaultFieldClasses.
UniCache (see page 161)	This is constant IBCustomDataSet.UniCache.

4.5.1 IBCustomDataSet.BufferCacheSize Constant

Pascal

```
BufferCacheSize = 1000;
```

C++

```
BufferCacheSize = 1000;
```

File

IBCustomDataSet

Description

This is constant IBCustomDataSet.BufferCacheSize.

4.5.2 IBCustomDataSet.DefaultFieldClasses Constant

Pascal

```
DefaultFieldClasses: array[TFieldType] of TFieldClass = ( nil, TIBStringField, TSmallintField,
TIntegerField, TWordField, TBooleanField, TFloatField, TCurrencyField, TIBBCDField,
TDateField, TTimeField, TDateTimeField, TBytesField, TVarBytesField, TAutoIncField,
TBlobField, TMemoField, TGraphicField, TBlobField, TBlobField, TBlobField, TBlobField, nil,
TStringField, nil, TLargeIntField, TADTField, TArrayField, TReferenceField, TDataSetField,
TBlobField, TMemoField, TVariantField, TInterfaceField, TIDispatchField, TGuidField,
TSQLTimeStampField, TFMTBcdField, nil, nil, nil, nil);
```

C++

```
array[TFieldType] of TFieldClass DefaultFieldClasses = ( nil, TIBStringField, TSmallintField,
TIntegerField, TWordField, TBooleanField, TFloatField, TCurrencyField, TIBBCDField,
TDateField, TTimeField, TDateTimeField, TBytesField, TVarBytesField, TAutoIncField,
TBlobField, TMemoField, TGraphicField, TBlobField, TBlobField, TBlobField, TBlobField, nil,
TStringField, nil, TLargeIntField, TADTField, TArrayField, TReferenceField, TDataSetField,
TBlobField, TMemoField, TVariantField, TInterfaceField, TIDispatchField, TGuidField,
TSQLTimeStampField, TFMTBcdField, nil, nil, nil, nil);
```

File

IBCustomDataSet

Description

This is constant IBCustomDataSet.DefaultFieldClasses.

4.5.3 IBCustomDataSet.UniCache Constant

Pascal

```
UniCache = 2;
```

C++

```
UniCache = 2;
```

File

IBCustomDataSet

Description

This is constant IBCustomDataSet.UniCache.

5 IBDatabase Namespace

This is namespace IBDatabase.

Topics

Name	Description
Classes (see page 165)	The following table lists classes in this documentation.
Interfaces (see page 221)	The following table lists interfaces in this documentation.
Functions (see page 224)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 225)	The following table lists structs, records, enums in this documentation.
Types (see page 227)	The following table lists types in this documentation.
Constants (see page 229)	The following table lists constants in this documentation.

Classes

	Name	Description
	TIBBase (see page 165)	TIBBase is the ancestor object from which TIBTransaction (see page 206) descends.
	TIBDataBase (see page 176)	TIBDatabase encapsulates an InterBase database connection.
	TIBSchema (see page 201)	This is class IBDatabase.TIBSchema.
	TIBTimer (see page 202)	This is class IBDatabase.TIBTimer.
	TIBTransaction (see page 206)	TIBTransaction provides discrete transaction control over a one or more database connections in a database application.

Constants

Name	Description
DPBConstantNames (see page 229)	This is constant IBDatabase.DPBConstantNames.
DPBPrefix (see page 230)	This is constant IBDatabase.DPBPrefix.
TPBConstantNames (see page 230)	This is constant IBDatabase.TPBConstantNames.
TPBPrefix (see page 230)	This is constant IBDatabase.TPBPrefix.

Functions

	Name	Description
	GenerateDPB (see page 224)	This is function IBDatabase.GenerateDPB.
	GenerateTPB (see page 224)	This is function IBDatabase.GenerateTPB.

Interfaces

	Name	Description
	IIBEventNotifier (see page 221)	IIBEventNotifier allows a TIBDataBase (see page 176) component to automatically register interest in events on the InterBase server.

Structs, Records, Enums

	Name	Description
	TAutoStopAction (see page 225)	TAutoStopAction indicates the action to take on automatically ended transactions.
	TTransactionAction (see page 226)	This is record IBDatabase.TTransactionAction.

Types

Name	Description
TIBDatabaseLoginEvent (see page 227)	TIBDatabaseLoginEvent is the type of event handlers that respond when an IB (see page 1) database component forms a connection to an InterBase server.
TIBFileName (see page 227)	This is type IBDatabase.TIBFileName.

5.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBBase (see page 165)	TIBBase is the ancestor object from which TIBTransaction (see page 206) descends.
	TIBDataBase (see page 176)	TIBDatabase encapsulates an InterBase database connection.
	TIBSchema (see page 201)	This is class IBDatabase.TIBSchema.
	TIBTimer (see page 202)	This is class IBDatabase.TIBTimer.
	TIBTransaction (see page 206)	TIBTransaction provides discrete transaction control over a one or more database connections in a database application.

5.1.1 TIBBase Class

TIBBase is the ancestor object from which TIBTransaction (see page 206) descends.

Class Hierarchy

Pascal

```
TIBBase = class(TObject);
```

C++

```
class TIBBase : public TObject;
```

File

IBDatabase

Description

TIBBase is the ancestor object from which TIBTransaction (see page 206) descends.

5.1.1.1 TIBBase Members

The following tables list the members exposed by TIBBase.

Protected Fields

	Name	Description
	FAfterDatabaseDisconnect (see page 167)	This is FAfterDatabaseDisconnect, a member of class TIBBase.
	FAfterTransactionEnd (see page 167)	This is FAfterTransactionEnd, a member of class TIBBase.
	FBeforeDatabaseDisconnect (see page 168)	This is FBeforeDatabaseDisconnect, a member of class TIBBase.
	FBeforeTransactionEnd (see page 168)	This is FBeforeTransactionEnd, a member of class TIBBase.
	FDatabase (see page 168)	This is FDatabase, a member of class TIBBase.
	FIndexInDatabase (see page 168)	This is FIndexInDatabase, a member of class TIBBase.
	FIndexInTransaction (see page 169)	This is FIndexInTransaction, a member of class TIBBase.
	FOnDatabaseFree (see page 169)	This is FOnDatabaseFree, a member of class TIBBase.

	FOnTransactionFree (see page 169)	This is FOnTransactionFree, a member of class TIBBase.
	FOwner (see page 169)	This is FOwner, a member of class TIBBase.
	FTransaction (see page 169)	This is FTransaction, a member of class TIBBase.

Protected Methods

	Name	Description
	DoAfterDatabaseDisconnect (see page 171)	This is DoAfterDatabaseDisconnect, a member of class TIBBase.
	DoAfterTransactionEnd (see page 171)	This is DoAfterTransactionEnd, a member of class TIBBase.
	DoBeforeDatabaseDisconnect (see page 171)	This is DoBeforeDatabaseDisconnect, a member of class TIBBase.
	DoBeforeTransactionEnd (see page 171)	This is DoBeforeTransactionEnd, a member of class TIBBase.
	DoDatabaseFree (see page 171)	This is DoDatabaseFree, a member of class TIBBase.
	DoTransactionFree (see page 172)	This is DoTransactionFree, a member of class TIBBase.
	GetDBHandle (see page 172)	This is GetDBHandle, a member of class TIBBase.
	GetTRHandle (see page 172)	This is GetTRHandle, a member of class TIBBase.
	SetDatabase (see page 172)	This is SetDatabase, a member of class TIBBase.
	SetTransaction (see page 173)	This is SetTransaction, a member of class TIBBase.

Public Events

	Name	Description
	AfterDatabaseDisconnect (see page 175)	Occurs after a database is disconnected.
	AfterTransactionEnd (see page 175)	Occurs after a transaction has ended.
	BeforeDatabaseDisconnect (see page 175)	Occurs before a database disconnects.
	BeforeTransactionEnd (see page 175)	Occurs before a transaction ends.
	OnDatabaseFree (see page 176)	Occurs after a database is freed from memory.
	OnTransactionFree (see page 176)	Occurs after a transaction has been freed from memory.

Public Methods

	Name	Description
	CheckDatabase (see page 170)	Checks if the database is active.
	CheckTransaction (see page 170)	Checks if the transaction is active.
	Create (see page 166)	Creates an instance of an IBBase object.
	Destroy (see page 167)	Destroys an IBBase object.

Public Properties

	Name	Description
	Database (see page 173)	Sets or returns the associated database.
	DBHandle (see page 173)	Indicates the database handle.
	Owner (see page 174)	Indicates which component owns the component.
	Transaction (see page 174)	Sets or returns the associated transaction.
	TRHandle (see page 174)	Returns the transaction handle.

5.1.1.2 TIBBase.Create Constructor

Creates an instance of an IBBase object.

Pascal

```
constructor Create(AOwner: TObject);
```

C++

```
__fastcall TIBBase(TObject * AOwner);
```

Description

Call Create to create an instance of an IBBase object.

5.1.1.3 TIBBase.Destroy Destructor

Destroys an IBBase object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBBase();
```

Description

Call Destroy to destroy an IBBase object and to free up any resources associated with it.

5.1.1.4 TIBBase Fields

The fields of the TIBBase class are listed here.

Protected Fields

	Name	Description
	FAfterDatabaseDisconnect (see page 167)	This is FAfterDatabaseDisconnect, a member of class TIBBase.
	FAfterTransactionEnd (see page 167)	This is FAfterTransactionEnd, a member of class TIBBase.
	FBeforeDatabaseDisconnect (see page 168)	This is FBeforeDatabaseDisconnect, a member of class TIBBase.
	FBeforeTransactionEnd (see page 168)	This is FBeforeTransactionEnd, a member of class TIBBase.
	FDatabase (see page 168)	This is FDatabase, a member of class TIBBase.
	FIndexInDatabase (see page 168)	This is FIndexInDatabase, a member of class TIBBase.
	FIndexInTransaction (see page 169)	This is FIndexInTransaction, a member of class TIBBase.
	FOnDatabaseFree (see page 169)	This is FOnDatabaseFree, a member of class TIBBase.
	FOnTransactionFree (see page 169)	This is FOnTransactionFree, a member of class TIBBase.
	FOwner (see page 169)	This is FOwner, a member of class TIBBase.
	FTransaction (see page 169)	This is FTransaction, a member of class TIBBase.

5.1.1.4.1 TIBBase.FAfterDatabaseDisconnect Field

Pascal

```
FAfterDatabaseDisconnect: TNotifyEvent;
```

C++

```
TNotifyEvent FAfterDatabaseDisconnect;
```

Description

This is FAfterDatabaseDisconnect, a member of class TIBBase.

5.1.1.4.2 TIBBase.FAfterTransactionEnd Field

Pascal

```
FAfterTransactionEnd: TNotifyEvent;
```

C++

```
TNotifyEvent FAfterTransactionEnd;
```

Description

This is FAfterTransactionEnd, a member of class TIBBase.

5.1.1.4.3 TIBBase.FBeforeDatabaseDisconnect Field

Pascal

```
FBeforeDatabaseDisconnect: TNotifyEvent;
```

C++

```
TNotifyEvent FBeforeDatabaseDisconnect;
```

Description

This is FBeforeDatabaseDisconnect, a member of class TIBBase.

5.1.1.4.4 TIBBase.FBeforeTransactionEnd Field

Pascal

```
FBeforeTransactionEnd: TNotifyEvent;
```

C++

```
TNotifyEvent FBeforeTransactionEnd;
```

Description

This is FBeforeTransactionEnd, a member of class TIBBase.

5.1.1.4.5 TIBBase.FDatabase Field

Pascal

```
FDatabase: TIBDatabase;
```

C++

```
TIBDatabase FDatabase;
```

Description

This is FDatabase, a member of class TIBBase.

5.1.1.4.6 TIBBase.FIndexInDatabase Field

Pascal

```
FIndexInDatabase: Integer;
```

C++

```
int FIndexInDatabase;
```

Description

This is FIndexInDatabase, a member of class TIBBase.

5.1.1.4.7 TIBBase.FIndexInTransaction Field

Pascal

```
FIndexInTransaction: Integer;
```

C++

```
int FIndexInTransaction;
```

Description

This is FIndexInTransaction, a member of class TIBBase.

5.1.1.4.8 TIBBase.FOnDatabaseFree Field

Pascal

```
FOnDatabaseFree: TNotifyEvent;
```

C++

```
TNotifyEvent FOnDatabaseFree;
```

Description

This is FOnDatabaseFree, a member of class TIBBase.

5.1.1.4.9 TIBBase.FOnTransactionFree Field

Pascal

```
FOnTransactionFree: TNotifyEvent;
```

C++

```
TNotifyEvent FOnTransactionFree;
```

Description

This is FOnTransactionFree, a member of class TIBBase.

5.1.1.4.10 TIBBase.FOwner Field

Pascal

```
FOwner: TObject;
```

C++

```
TObject * FOwner;
```

Description

This is FOwner, a member of class TIBBase.

5.1.1.4.11 TIBBase.FTransaction Field

Pascal

```
FTransaction: TIBTransaction;
```

C++

```
TIBTransaction FTransaction;
```

Description

This is FTransaction, a member of class TIBBase.

5.1.1.5 TIBBase Methods

The methods of the TIBBase class are listed here.

Protected Methods

	Name	Description
	DoAfterDatabaseDisconnect (see page 171)	This is DoAfterDatabaseDisconnect, a member of class TIBBase.
	DoAfterTransactionEnd (see page 171)	This is DoAfterTransactionEnd, a member of class TIBBase.
	DoBeforeDatabaseDisconnect (see page 171)	This is DoBeforeDatabaseDisconnect, a member of class TIBBase.
	DoBeforeTransactionEnd (see page 171)	This is DoBeforeTransactionEnd, a member of class TIBBase.
	DoDatabaseFree (see page 171)	This is DoDatabaseFree, a member of class TIBBase.
	DoTransactionFree (see page 172)	This is DoTransactionFree, a member of class TIBBase.
	GetDBHandle (see page 172)	This is GetDBHandle, a member of class TIBBase.
	GetTRHandle (see page 172)	This is GetTRHandle, a member of class TIBBase.
	SetDatabase (see page 172)	This is SetDatabase, a member of class TIBBase.
	SetTransaction (see page 173)	This is SetTransaction, a member of class TIBBase.

Public Methods

	Name	Description
	CheckDatabase (see page 170)	Checks if the database is active.
	CheckTransaction (see page 170)	Checks if the transaction is active.

5.1.1.5.1 TIBBase.CheckDatabase Method

Checks if the database is active.

Pascal

```
procedure CheckDatabase; virtual;
```

C++

```
virtual __fastcall CheckDatabase();
```

Description

Call CheckDatabase to check if the database is assigned and active.

5.1.1.5.2 TIBBase.CheckTransaction Method

Checks if the transaction is active.

Pascal

```
procedure CheckTransaction; virtual;
```

C++

```
virtual __fastcall CheckTransaction();
```

Description

Call CheckTransaction to check if the transaction is assigned and active.

5.1.1.5.3 TIBBase.DoAfterDatabaseDisconnect Method

Pascal

```
procedure DoAfterDatabaseDisconnect; virtual;
```

C++

```
virtual __fastcall DoAfterDatabaseDisconnect();
```

Description

This is DoAfterDatabaseDisconnect, a member of class TIBBase.

5.1.1.5.4 TIBBase.DoAfterTransactionEnd Method

Pascal

```
procedure DoAfterTransactionEnd; virtual;
```

C++

```
virtual __fastcall DoAfterTransactionEnd();
```

Description

This is DoAfterTransactionEnd, a member of class TIBBase.

5.1.1.5.5 TIBBase.DoBeforeDatabaseDisconnect Method

Pascal

```
procedure DoBeforeDatabaseDisconnect; virtual;
```

C++

```
virtual __fastcall DoBeforeDatabaseDisconnect();
```

Description

This is DoBeforeDatabaseDisconnect, a member of class TIBBase.

5.1.1.5.6 TIBBase.DoBeforeTransactionEnd Method

Pascal

```
procedure DoBeforeTransactionEnd; virtual;
```

C++

```
virtual __fastcall DoBeforeTransactionEnd();
```

Description

This is DoBeforeTransactionEnd, a member of class TIBBase.

5.1.1.5.7 TIBBase.DoDatabaseFree Method

Pascal

```
procedure DoDatabaseFree; virtual;
```

C++

```
virtual __fastcall DoDatabaseFree();
```

Description

This is DoDatabaseFree, a member of class TIBBase.

5.1.1.5.8 TIBBase.DoTransactionFree Method

Pascal

```
procedure DoTransactionFree; virtual;
```

C++

```
virtual __fastcall DoTransactionFree();
```

Description

This is DoTransactionFree, a member of class TIBBase.

5.1.1.5.9 TIBBase.GetDBHandle Method

Pascal

```
function GetDBHandle: PISC_DB_HANDLE; virtual;
```

C++

```
virtual __fastcall PISC_DB_HANDLE GetDBHandle();
```

Description

This is GetDBHandle, a member of class TIBBase.

5.1.1.5.10 TIBBase.GetTRHandle Method

Pascal

```
function GetTRHandle: PISC_TR_HANDLE; virtual;
```

C++

```
virtual __fastcall PISC_TR_HANDLE GetTRHandle();
```

Description

This is GetTRHandle, a member of class TIBBase.

5.1.1.5.11 TIBBase.SetDatabase Method

Pascal

```
procedure SetDatabase(Value: TIBDatabase); virtual;
```

C++

```
virtual __fastcall SetDatabase(TIBDatabase Value);
```

Description

This is SetDatabase, a member of class TIBBase.

5.1.1.5.12 TIBBase.SetTransaction Method

Pascal

```
procedure SetTransaction(Value: TIBTransaction); virtual;
```

C++

```
virtual __fastcall SetTransaction(TIBTransaction Value);
```

Description

This is SetTransaction, a member of class TIBBase.

5.1.1.6 TIBBase Properties

The properties of the TIBBase class are listed here.

Public Properties

	Name	Description
	Database (see page 173)	Sets or returns the associated database.
R	DBHandle (see page 173)	Indicates the database handle.
R	Owner (see page 174)	Indicates which component owns the component.
	Transaction (see page 174)	Sets or returns the associated transaction.
R	TRHandle (see page 174)	Returns the transaction handle.

5.1.1.6.1 TIBBase.Database Property

Sets or returns the associated database.

Pascal

```
property Database: TIBDatabase;
```

C++

```
__property TIBDatabase Database;
```

Description

Use Database to set or return the associated database.

5.1.1.6.2 TIBBase.DBHandle Property

Indicates the database handle.

Pascal

```
property DBHandle: PISC_DB_HANDLE;
```

C++

```
__property PISC_DB_HANDLE DBHandle;
```

Description

Use DBHandle to return the database handle.

5.1.1.6.3 TIBBase.Owner Property

Indicates which component owns the component.

Pascal

```
property Owner: TObject;
```

C++

```
__property TObject * Owner;
```

Description

Owner refers to the SQL object (DataSet, TIBSQL, or Blob) that created the TIBBase (see page 165) component.

5.1.1.6.4 TIBBase.Transaction Property

Sets or returns the associated transaction.

Pascal

```
property Transaction: TIBTransaction;
```

C++

```
__property TIBTransaction Transaction;
```

Description

Use Transaction to set or return the associated transaction.

5.1.1.6.5 TIBBase.TRHandle Property

Returns the transaction handle.

Pascal

```
property TRHandle: PISC_TR_HANDLE;
```

C++

```
__property PISC_TR_HANDLE TRHandle;
```

Description

Use TRHandle to return the transaction handle.

5.1.1.7 TIBBase Events

The events of the TIBBase class are listed here.

Public Events

	Name	Description
	AfterDatabaseDisconnect (see page 175)	Occurs after a database is disconnected.
	AfterTransactionEnd (see page 175)	Occurs after a transaction has ended.
	BeforeDatabaseDisconnect (see page 175)	Occurs before a database disconnects.
	BeforeTransactionEnd (see page 175)	Occurs before a transaction ends.
	OnDatabaseFree (see page 176)	Occurs after a database is freed from memory.
	OnTransactionFree (see page 176)	Occurs after a transaction has been freed from memory.

5.1.1.7.1 TIBBase.AfterDatabaseDisconnect Event

Occurs after a database is disconnected.

Pascal

```
property AfterDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterDatabaseDisconnect;
```

Description

Write an AfterDatabaseDisconnect event handler to take specific actions after a database is disconnected.

5.1.1.7.2 TIBBase.AfterTransactionEnd Event

Occurs after a transaction has ended.

Pascal

```
property AfterTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterTransactionEnd;
```

Description

Write an AfterTransactionEnd event handler to take specific actions after a transaction has ended.

5.1.1.7.3 TIBBase.BeforeDatabaseDisconnect Event

Occurs before a database disconnects.

Pascal

```
property BeforeDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeDatabaseDisconnect;
```

Description

Write a BeforeDatabaseDisconnect event handler to take specific actions before a database disconnects.

5.1.1.7.4 TIBBase.BeforeTransactionEnd Event

Occurs before a transaction ends.

Pascal

```
property BeforeTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeTransactionEnd;
```

Description

Write a BeforeTransactionEnd event handler to take specific actions before a transaction ends.

5.1.1.7.5 TIBBase.OnDatabaseFree Event

Occurs after a database is freed from memory.

Pascal

```
property OnDatabaseFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnDatabaseFree;
```

Description

Write an OnDatabaseFree event handler to take specific actions after a database is freed from memory.

5.1.1.7.6 TIBBase.OnTransactionFree Event

Occurs after a transaction has been freed from memory.

Pascal

```
property OnTransactionFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnTransactionFree;
```

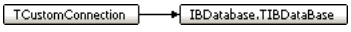
Description

Write an OnTransactionFree event handler to take specific actions after a transaction has been freed from memory.

5.1.2 TIBDataBase Class

TIBDatabase encapsulates an InterBase database connection.

Class Hierarchy



Pascal

```
TIBDataBase = class(TCustomConnection);
```

C++

```
class TIBDataBase : public TCustomConnection;
```

File

IBDatabase

Description

Use TIBDataBase to encapsulate an InterBase database connection. All TIBCustomDataSet descendants and TIBSQL use the TIBDataBase component to gain access to databases.

See Also

Connecting to Databases: Overview

Connecting Directly to a Database Server

Getting Started with InterBase Express

5.1.2.1 TIBDataBase Members

The following tables list the members exposed by TIBDataBase.

Protected Methods

	Name	Description
	DoConnect (see page 182)	Provides the interface for a method that opens a connection.
	DoDisconnect (see page 183)	Provides the interface for a method that terminates the connection.
	GetConnected (see page 184)	Returns the value of the Connected property.
	Loaded (see page 186)	Initializes the connection component after all objects in the form or data module have been loaded.
	Notification (see page 187)	Forwards notification messages to all owned components.

Public Methods

	Name	Description
	AddEventNotifier (see page 180)	Adds an event notifier to the database's internal list so that the database can handle registering and unregistering interest in events.
	AddTransaction (see page 180)	Adds an association between the transaction component and the database component.
	ApplyUpdates (see page 180)	Posts pending cached updates for specified datasets to the database server.
	Call (see page 181)	Returns an error message based on the error code.
	CheckActive (see page 181)	Checks to see if the database connection is active.
	CheckDatabaseName (see page 181)	Checks to see if the DatabaseName (see page 190) property is not empty.
	CheckInactive (see page 181)	Checks to see if the database connection is inactive.
	CloseDataSets (see page 182)	Closes all datasets associated with the database component without disconnecting from the database server.
	Create (see page 178)	Creates an instance of a TIBDataBase (see page 176) component.
	CreateDatabase (see page 182)	Creates a database using Params (see page 196).
	Destroy (see page 178)	Destroys the instance of the database component.
	DropDatabase (see page 183)	Drops a database.
	FindDefaultTransaction (see page 183)	Returns the default transaction.
	FindTransaction (see page 184)	Finds the index of a transaction.
	FlushSchema (see page 184)	Clears the schema cache for the database.
	ForceClose (see page 184)	Forces the database connection to close.
	GetFieldNames (see page 185)	Populates a list with the names of the fields in the table.
	GetTableNames (see page 185)	Populates a string list with the names of tables in the database.
	Has_COMPUTED_BLR (see page 185)	Indicates whether a specified field in a specified relation includes a COMPUTED_BLR attribute.
	Has_DEFAULT_VALUE (see page 186)	Indicates whether a specified field in a specified relation includes a DEFAULT_VALUE attribute.
	IndexOfDBConst (see page 186)	Searches for the named parameter in the database parameters list.
	RemoveEventNotifier (see page 187)	Removes an event notifier from the database's internal list so that the database no longer handles registering and unregistering interest in events.
	RemoveTransaction (see page 187)	Disassociates a transaction from the database.
	RemoveTransactions (see page 188)	Disassociates all transactions from the database.
	SetHandle (see page 188)	Sets the handle for the database.
	TestConnected (see page 188)	Tests whether a database is connected.

Public Properties

	Name	Description
	DBParamByDPB (see page 190)	Specifies the name of the database to associate with this database component.
	Handle (see page 191)	Specifies the InterBase API database handle.
	HandleIsShared (see page 191)	Indicates whether or not a the handle is shared.
	InternalTransaction (see page 192)	Used internally by InterBase transactions.
	IsReadOnly (see page 192)	Indicates whether or not the database is set to read-only.

	SQLObjectCount (see page 197)	Returns the number of SQL objects.
	SQLObjects (see page 197)	Returns an SQL object.
	TransactionCount (see page 198)	Returns the number of transactions associated with the TIBDataBase (see page 176) component.
	Transactions (see page 198)	Specifies a transaction for the given index.

Published Events

	Name	Description
	AfterConnect (see page 198)	Occurs after a connection is established.
	AfterDisconnect (see page 199)	Occurs after the connection closes.
	BeforeConnect (see page 199)	Occurs immediately before establishing a connection.
	BeforeDisconnect (see page 199)	Occurs immediately before the connection closes.
	OnDialectDowngradeWarning (see page 200)	Occurs after the SQL dialect of the client connection is downgraded.
	OnIdleTimer (see page 200)	Occurs after a database connection times out.
	OnLogin (see page 201)	Occurs when an application connects to a database.

Published Properties

	Name	Description
	AllowStreamedConnected (see page 189)	Indicates whether the Connected (see page 189) property can be set at design time.
	Connected (see page 189)	Indicates whether or not a database connection is active.
	DatabaseName (see page 190)	Specifies the name of the database to associate with this database component.
	DBSQLDialect (see page 190)	Returns the database SQL dialect.
	DefaultTransaction (see page 191)	Sets or returns the default database transaction.
	IdleTimer (see page 191)	Specifies how long the database should wait before disconnecting an idle connection.
	LoginPrompt (see page 192)	Specifies whether a login dialog appears immediately before opening a new connection.
	Params (see page 196)	Specifies the database parameters to pass to the InterBase server.
	SQLDialect (see page 196)	Sets or returns the SQL dialect used by the client.
	TraceFlags (see page 197)	Specifies the database operations to track with the SQL Monitor at runtime.

5.1.2.2 TIBDataBase.Create Constructor

Creates an instance of a TIBDataBase (see page 176) component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBDataBase(TComponent * AOwner);
```

Description

Call (see page 181) Create to instantiate a database component at runtime. An application can create a database component in order to control the component's existence and set its properties and events.

Create instantiates a database component and creates an empty list of dataset components for the DataSets property and an empty string list for the Params (see page 196) property.

5.1.2.3 TIBDataBase.Destroy Destructor

Destroys the instance of the database component.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBDataBase();
```

Description

Do not call Destroy directly in an application. Instead, call Free, which verifies that the database component is not already freed before calling Destroy.

Destroy disconnects from the database server, if necessary. It then frees the string resources allocated for the Params (see page 196) and DataSets properties before calling its inherited destructor.

5.1.2.4 TIBDataBase Methods

The methods of the TIBDataBase class are listed here.

Protected Methods

	Name	Description
	DoConnect (see page 182)	Provides the interface for a method that opens a connection.
	DoDisconnect (see page 183)	Provides the interface for a method that terminates the connection.
	GetConnected (see page 184)	Returns the value of the Connected property.
	Loaded (see page 186)	Initializes the connection component after all objects in the form or data module have been loaded.
	Notification (see page 187)	Forwards notification messages to all owned components.

Public Methods

	Name	Description
	AddEventNotifier (see page 180)	Adds an event notifier to the database's internal list so that the database can handle registering and unregistering interest in events.
	AddTransaction (see page 180)	Adds an association between the transaction component and the database component.
	ApplyUpdates (see page 180)	Posts pending cached updates for specified datasets to the database server.
	Call (see page 181)	Returns an error message based on the error code.
	CheckActive (see page 181)	Checks to see if the database connection is active.
	CheckDatabaseName (see page 181)	Checks to see if the DatabaseName (see page 190) property is not empty.
	CheckInactive (see page 181)	Checks to see if the database connection is inactive.
	CloseDataSets (see page 182)	Closes all datasets associated with the database component without disconnecting from the database server.
	CreateDatabase (see page 182)	Creates a database using Params (see page 196).
	DropDatabase (see page 183)	Drops a database.
	FindDefaultTransaction (see page 183)	Returns the default transaction.
	FindTransaction (see page 184)	Finds the index of a transaction.
	FlushSchema (see page 184)	Clears the schema cache for the database.
	ForceClose (see page 184)	Forces the database connection to close.
	GetFieldNames (see page 185)	Populates a list with the names of the fields in the table.
	GetTableNames (see page 185)	Populates a string list with the names of tables in the database.
	Has_COMPUTED_BLR (see page 185)	Indicates whether a specified field in a specified relation includes a COMPUTED_BLR attribute.
	Has_DEFAULT_VALUE (see page 186)	Indicates whether a specified field in a specified relation includes a DEFAULT_VALUE attribute.
	IndexOfDBConst (see page 186)	Searches for the named parameter in the database parameters list.
	RemoveEventNotifier (see page 187)	Removes an event notifier from the database's internal list so that the database no longer handles registering and unregistering interest in events.
	RemoveTransaction (see page 187)	Disassociates a transaction from the database.

	RemoveTransactions (see page 188)	Disassociates all transactions from the database.
	SetHandle (see page 188)	Sets the handle for the database.
	TestConnected (see page 188)	Tests whether a database is connected.

5.1.2.4.1 TIBDataBase.AddEventNotifier Method

Adds an event notifier to the database's internal list so that the database can handle registering and unregistering interest in events.

Pascal

```
procedure AddEventNotifier(Notifier: IIBEventNotifier);
```

C++

```
__fastcall AddEventNotifier(IIBEventNotifier Notifier);
```

Description

Call (see page 181) AddEventNotifier when you want the database to automatically register and unregister an interest in server events when it opens and closes a connection to the server.

Notifier is the interface to an object that has in interest in a set of events on the InterBase server. For example, Notifier can be the interface to a TIBEvents object.

AddEventNotifier adds Notifier to its internal list of event notifiers. When TIBDataBase (see page 176) opens a connection, it checks every notifier in its list, and if a notifier indicates that it wants to be auto-registered, AddEventNotifier calls its RegisterEvents method so that the notifier registers its interest with the InterBase Event manager. Before TIBDataBase (see page 176) closes a connection, it unregisters interest from every notifier in its list.

Note: If Notifier is a TIBEvents object, AddEventNotifier is automatically called when you set the Database property.

5.1.2.4.2 TIBDataBase.AddTransaction Method

Adds an association between the transaction component and the database component.

Pascal

```
function AddTransaction(TR: TIBTransaction): Integer;
```

C++

```
__fastcall int AddTransaction(TIBTransaction TR);
```

Description

Call (see page 181) AddTransaction to add an association between the transaction component and the database component.

5.1.2.4.3 TIBDataBase.ApplyUpdates Method

Posts pending cached updates for specified datasets to the database server.

Pascal

```
procedure ApplyUpdates(const DataSets: array of TDataSet);
```

C++

```
__fastcall ApplyUpdates(const array of TDataSet DataSets);
```

Description

Call (see page 181) ApplyUpdates to post pending cached updates for a specific set of open datasets to the database server. ApplyUpdates is only meaningful if the CachedUpdates property of a specified dataset is true.

DataSets is a list of dataset names specifying the datasets for which to post pending updates. DataSets need not list every currently open dataset. For each listed dataset ApplyUpdates calls the dataset's ApplyUpdates and CommitUpdates methods to post that dataset's pending cached updates.

5.1.2.4.4 TIBDataBase.Call Method

Returns an error message based on the error code.

Pascal

```
function Call(ErrCode: ISC_STATUS; RaiseError: Boolean): ISC_STATUS;
```

C++

```
__fastcall ISC_STATUS Call(ISC_STATUS ErrCode, Boolean RaiseError);
```

Description

Call is an internal method used to make calls to the InterBase API, and gives you the option of raising an exception or returning an error based on the value of RaiseError.

5.1.2.4.5 TIBDataBase.CheckActive Method

Checks to see if the database connection is active.

Pascal

```
procedure CheckActive;
```

C++

```
__fastcall CheckActive();
```

Description

Call (see page 181) CheckActive to determine if the connection to a database server is active. If the database connection is inactive, an ibxeDatabaseClosed exception is raised.

5.1.2.4.6 TIBDataBase.CheckDatabaseName Method

Checks to see if the DatabaseName (see page 190) property is not empty.

Pascal

```
procedure CheckDatabaseName;
```

C++

```
__fastcall CheckDatabaseName();
```

Description

Call (see page 181) CheckDatabaseName to determine whether the DatabaseName (see page 190) property is empty. If the DatabaseName (see page 190) is empty, an ibxeDatabaseNameMissing exception is raised.

5.1.2.4.7 TIBDataBase.CheckInactive Method

Checks to see if the database connection is inactive.

Pascal

```
procedure CheckInactive;
```

C++

```
__fastcall CheckInactive();
```

Description

Call (see page 181) CheckInactive to determine whether the connection to a database server is inactive. If the database connection is active, an ibxeDatabaseOpen exception is raised.

5.1.2.4.8 TIBDataBase.CloseDataSets Method

Closes all datasets associated with the database component without disconnecting from the database server.

Pascal

```
procedure CloseDataSets;
```

C++

```
__fastcall CloseDataSets();
```

Description

Call (see page 181) CloseDataSets to close all active datasets without disconnecting from the database server. Ordinarily, when an application calls Close, all datasets are closed, and the connection to the database server is dropped. Calling CloseDataSets instead of Close ensures that an application can close all active datasets without having to reconnect to the database server at a later time.

5.1.2.4.9 TIBDataBase.CreateDatabase Method

Creates a database using Params (see page 196).

Pascal

```
procedure CreateDatabase;
```

C++

```
__fastcall CreateDatabase();
```

Description

Call (see page 181) CreateDatabase to create a database using Params (see page 196) as the rest of the CREATE DATABASE command.

For example, if you wanted to create a local InterBase database, you could do the following:

1. Set the database name to the drive, path, and filename of the database file.
2. Set Params (see page 196) to the parameter for the CREATE DATABASE statement: `USER "SYSDBA" PASSWORD "masterkey" PAGE_SIZE 4096`
3. Set the SQLDialect (see page 196) value.
4. Call (see page 181) the CreateDatabase method.

5.1.2.4.10 TIBDataBase.DoConnect Method

Provides the interface for a method that opens a connection.

Pascal

```
procedure DoConnect; override;
```

C++

```
virtual __fastcall DoConnect();
```

Description

The Connected property uses DoConnect to establish a connection. Descendant classes override the DoConnect method to establish their connection as appropriate. As implemented in TCustomConnection, DoConnect does nothing.

See Also

Connected

DoDisconnect

5.1.2.4.11 TIBDataBase.DoDisconnect Method

Provides the interface for a method that terminates the connection.

Pascal

```
procedure DoDisconnect; override;
```

C++

```
virtual __fastcall DoDisconnect();
```

Description

The Connected property uses DoDisconnect to close a connection. Descendant classes override the DoDisconnect method to drop a connection. As implemented in TCustomConnection, DoDisconnect does nothing.

See Also

Connected

DoConnect

5.1.2.4.12 TIBDataBase.DropDatabase Method

Drops a database.

Pascal

```
procedure DropDatabase;
```

C++

```
__fastcall DropDatabase();
```

Description

Call (see page 181) DropDatabase to drop a database, which removes the database file from the server.

5.1.2.4.13 TIBDataBase.FindDefaultTransaction Method

Returns the default transaction.

Pascal

```
function FindDefaultTransaction: TIBTransaction;
```

C++

```
__fastcall TIBTransaction FindDefaultTransaction();
```

Description

Call (see page 181) FindDefaultTransaction to return the default transaction for the database.

5.1.2.4.14 TIBDataBase.FindTransaction Method

Finds the index of a transaction.

Pascal

```
function FindTransaction(TR: TIBTransaction): Integer;
```

C++

```
__fastcall int FindTransaction(TIBTransaction TR);
```

Description

Call (see page 181) FindTransaction to find the index of a specified transaction.

5.1.2.4.15 TIBDataBase.FlushSchema Method

Clears the schema cache for the database.

Pascal

```
procedure FlushSchema;
```

C++

```
__fastcall FlushSchema();
```

Description

Call (see page 181) FlushSchema to clear the schema cache.

5.1.2.4.16 TIBDataBase.ForceClose Method

Forces the database connection to close.

Pascal

```
procedure ForceClose;
```

C++

```
__fastcall ForceClose();
```

Description

Use ForceClose to force the database connection to close.

Note: Forcing a database to close attempts to close the connection to the server. Even if the call fails, the database handle is reset to nil (Delphi) or NULL (C++).

5.1.2.4.17 TIBDataBase.GetConnected Method

Returns the value of the Connected property.

Pascal

```
function GetConnected: Boolean; override;
```

C++

```
virtual __fastcall Boolean GetConnected();
```

Description

GetConnected is the protected read implementation of the Connected property. Descendant classes override this method to return true if a connection is open and false otherwise. As defined in TCustomConnection, GetConnected always returns false.

See Also

Connected

SetConnected

5.1.2.4.18 TIBDataBase.GetFieldNames Method

Populates a list with the names of the fields in the table.

Pascal

```
procedure GetFieldNames(const TableName: Widestring; List: TWideStrings);
```

C++

```
__fastcall GetFieldNames(const BSTR TableName, TWideStrings List);
```

Description

Call (see page 181) GetFieldNames to retrieve a list of fields in the associated table.

5.1.2.4.19 TIBDataBase.GetTableNames Method

Populates a string list with the names of tables in the database.

Pascal

```
procedure GetTableNames(List: TWideStrings; SystemTables: Boolean = False);
```

C++

```
__fastcall GetTableNames(TWideStrings List, Boolean SystemTables = False);
```

Description

Call (see page 181) GetTableNames to retrieve a list of tables in the associated database.

List is the already-existing string list object into which the tables' names are put.

Set SystemTables to indicate whether the list of table names should include the database's system tables.

```
IBDatabase1.GetTableNames(ListBox2.Items, False);
    IBDatabase1->GetTableNames(ListBox2->Items, false);
```

Note: Any contents already in the target string list object are eliminated and overwritten by the data produced by GetTableNames.

5.1.2.4.20 TIBDataBase.Has_COMPUTED_BLR Method

Indicates whether a specified field in a specified relation includes a COMPUTED_BLR attribute.

Pascal

```
function Has_COMPUTED_BLR(Relation: String; Field: String): Boolean;
```

C++

```
__fastcall Boolean Has_COMPUTED_BLR(UnicodeString Relation, UnicodeString Field);
```

Description

Call (see page 181) Has_COMPUTED_BLR to determine whether the Field in Relation includes a COMPUTED_BLR attribute.

5.1.2.4.21 TIBDataBase.Has_DEFAULT_VALUE Method

Indicates whether a specified field in a specified relation includes a DEFAULT_VALUE attribute.

Pascal

```
function Has_DEFAULT_VALUE(Relation: String; Field: String): Boolean;
```

C++

```
__fastcall Boolean Has_DEFAULT_VALUE(UnicodeString Relation, UnicodeString Field);
```

Description

Call (see page 181) Has_DEFAULT_VALUE to determine whether Field has a default value when used in Relation.

5.1.2.4.22 TIBDataBase.IndexOfDBConst Method

Searches for the named parameter in the database parameters list.

Pascal

```
function IndexOfDBConst(st: String): Integer;
```

C++

```
__fastcall int IndexOfDBConst(UnicodeString st);
```

Description

Use IndexOfDBConst to locate a parameter in the database parameters list. If the parameter is not found, IndexOfDBConst returns -1.

5.1.2.4.23 TIBDataBase.Loaded Method

Initializes the connection component after all objects in the form or data module have been loaded.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

Do not call the protected Loaded method. It is called automatically after the form or data module that contains the connection component has loaded all its owned objects. TCustomConnection overrides the inherited method to establish a connection if the Connected property is true.

See Also

StreamedConnected

Loaded

5.1.2.4.24 TIBDataBase.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

5.1.2.4.25 TIBDataBase.RemoveEventNotifier Method

Removes an event notifier from the database's internal list so that the database no longer handles registering and unregistering interest in events.

Pascal

```
procedure RemoveEventNotifier(Notifier: IIBEventNotifier);
```

C++

```
__fastcall RemoveEventNotifier(IIBEventNotifier Notifier);
```

Description

Call (see page 181) RemoveEventNotifier when you no longer want the database to automatically register and unregister an interest in server events when it opens and closes a connection to the server. For example, TIBEvents automatically calls RemoveEventNotifier when you change its Database property.

Notifier is the event notifier previously added to the database's internal list by a call to AddEventNotifier (see page 180).

RemoveEventNotifier removes Notifier from its internal list of event notifiers.

Note: Removing an event notifier from the list does not first unregister an interest in any registered events.

5.1.2.4.26 TIBDataBase.RemoveTransaction Method

Disassociates a transaction from the database.

Pascal

```
procedure RemoveTransaction(Idx: Integer);
```

C++

```
__fastcall RemoveTransaction(int Idx);
```

Description

Call (see page 181) RemoveTransaction to disassociate a specified transaction from the database.

5.1.2.4.27 TIBDataBase.RemoveTransactions Method

Disassociates all transactions from the database.

Pascal

```
procedure RemoveTransactions;
```

C++

```
__fastcall RemoveTransactions();
```

Description

Call (see page 181) RemoveTransactions to disassociate all transactions from the database.

5.1.2.4.28 TIBDataBase.SetHandle Method

Sets the handle for the database.

Pascal

```
procedure SetHandle(Value: TISC_DB_HANDLE);
```

C++

```
__fastcall SetHandle(TISC_DB_HANDLE Value);
```

Description

Call (see page 181) SetHandle to set the handle for the database.

5.1.2.4.29 TIBDataBase.TestConnected Method

Tests whether a database is connected.

Pascal

```
function TestConnected: Boolean;
```

C++

```
__fastcall Boolean TestConnected();
```

Description

Use TestConnected to determine whether a database is connected to the server. TestConnected returns true if the connection is good, and false if it is not.

5.1.2.5 TIBDataBase Properties

The properties of the TIBDataBase class are listed here.

Public Properties

	Name	Description
	DBParamByDPB (see page 190)	Specifies the name of the database to associate with this database component.
R	Handle (see page 191)	Specifies the InterBase API database handle.
R	HandleIsShared (see page 191)	Indicates whether or not a the handle is shared.
R	InternalTransaction (see page 192)	Used internally by InterBase transactions.
R	IsReadOnly (see page 192)	Indicates whether or not the database is set to read-only.
R	SQLObjectCount (see page 197)	Returns the number of SQL objects.
R	SQLObjects (see page 197)	Returns an SQL object.
R	TransactionCount (see page 198)	Returns the number of transactions associated with the TIBDataBase (see page 176) component.
R	Transactions (see page 198)	Specifies a transaction for the given index.

Published Properties

	Name	Description
	AllowStreamedConnected (see page 189)	Indicates whether the Connected (see page 189) property can be set at design time.
	Connected (see page 189)	Indicates whether or not a database connection is active.
	DatabaseName (see page 190)	Specifies the name of the database to associate with this database component.
R	DBSQLDialect (see page 190)	Returns the database SQL dialect.
	DefaultTransaction (see page 191)	Sets or returns the default database transaction.
	IdleTimer (see page 191)	Specifies how long the database should wait before disconnecting an idle connection.
	LoginPrompt (see page 192)	Specifies whether a login dialog appears immediately before opening a new connection.
	Params (see page 196)	Specifies the database parameters to pass to the InterBase server.
	SQLDialect (see page 196)	Sets or returns the SQL dialect used by the client.
	TraceFlags (see page 197)	Specifies the database operations to track with the SQL Monitor at runtime.

5.1.2.5.1 TIBDataBase.AllowStreamedConnected Property

Indicates whether the Connected (see page 189) property can be set at design time.

Pascal

```
property AllowStreamedConnected: Boolean;
```

C++

```
__property Boolean AllowStreamedConnected;
```

Description

When AllowStreamedConnected is true (the default), then setting the Connected (see page 189) property to true at design time causes this component to automatically establish a database connection when the application starts up. When AllowStreamedConnected is false, the connection must be explicitly opened at runtime, either by setting the Connected (see page 189) property to true, or by calling the Open method.

Set AllowStreamedConnected to false when using a local test database at design time and a different database at runtime. This allows the application to set the appropriate runtime connection information before opening a runtime connection, without having to explicitly shut down the database connection that is used at design time.

5.1.2.5.2 TIBDataBase.Connected Property

Indicates whether or not a database connection is active.

Pascal

```
property Connected;
```

C++

```
__property Connected;
```

Description

Set Connected to true to establish a database connection without opening a dataset. Set Connected to false to close a database connection. An application can check Connected to determine the current status of a database connection. If Connected is true, the database connection is active; if false, then the connection is inactive.

5.1.2.5.3 TIBDataBase.DatabaseName Property

Specifies the name of the database to associate with this database component.

Pascal

```
property DatabaseName: TIBFileName;
```

C++

```
__property TIBFileName DatabaseName;
```

Description

Use DatabaseName to specify the name of the database to use with a database component. For local InterBase databases, this can be a filename.

To connect to an InterBase database on a remote server using TCP/IP the syntax is <server_name>:<filename>.

To connect to an InterBase database on a remote server using NetBEUI, the syntax is: \\<server_name>\<filename>.

To connect to an InterBase database on a remote server using SPX, the syntax is: <server_name>@<filename>.

5.1.2.5.4 TIBDataBase.DBParamByDPB Property

Specifies the name of the database to associate with this database component.

Pascal

```
property DBParamByDPB [const Idx: Integer]: String;
```

C++

```
__property UnicodeString DBParamByDPB[int const Idx];
```

Description

Use DBParamByDPB to inspect and set DPB parameters without looking at the Params (see page 196) string list.

For example,

```
DBParamByDPB[isc_dpb_user_name]
```

can be used to set and inspect the user name.

5.1.2.5.5 TIBDataBase.DBSQLDialect Property

Returns the database SQL dialect.

Pascal

```
property DBSQLDialect: Integer;
```

C++

```
__property int DBSQLDialect;
```

Description

Use DBSQLDialect to get the database SQL dialect.

5.1.2.5.6 TIBDataBase.DefaultTransaction Property

Sets or returns the default database transaction.

Pascal

```
property DefaultTransaction: TIBTransaction;
```

C++

```
__property TIBTransaction DefaultTransaction;
```

Description

Use DefaultTransaction to set or return the default database transaction.

A single database connection can manage one or more transactions. DefaultTransaction is a convenient way to specify a default transaction to a database connection.

5.1.2.5.7 TIBDataBase.Handle Property

Specifies the InterBase API database handle.

Pascal

```
property Handle: TISC_DB_HANDLE;
```

C++

```
__property TISC_DB_HANDLE Handle;
```

Description

Use Handle to make calls directly to the InterBase API. Many of the InterBase API functions require a database handle as one of their arguments. Handle is assigned an initial value when a database is opened.

5.1.2.5.8 TIBDataBase.HandleIsShared Property

Indicates whether or not a the handle is shared.

Pascal

```
property HandleIsShared: Boolean;
```

C++

```
__property Boolean HandleIsShared;
```

Description

Read HandleIsShared to determine if the handle to the database is shared.

5.1.2.5.9 TIBDataBase.IdleTimer Property

Specifies how long the database should wait before disconnecting an idle connection.

Pascal

```
property IdleTimer: Integer;
```

C++

```
__property int IdleTimer;
```

Description

Use IdleTimer to indicate how long the database should wait before automatically terminating the connection.

5.1.2.5.10 TIBDataBase.InternalTransaction Property

Used internally by InterBase transactions.

Pascal

```
property InternalTransaction: TIBTransaction;
```

C++

```
__property TIBTransaction InternalTransaction;
```

Description

Do not use InternalTransaction directly. It is for internal use only.

5.1.2.5.11 TIBDataBase.IsReadOnly Property

Indicates whether or not the database is set to read-only.

Pascal

```
property IsReadOnly: Boolean;
```

C++

```
__property Boolean IsReadOnly;
```

Description

Read IsReadOnly to determine if the database is read-only.

Read-only databases are an InterBase 6 feature.

5.1.2.5.12 TIBDataBase.LoginPrompt Property

Specifies whether a login dialog appears immediately before opening a new connection.

Pascal

```
property LoginPrompt;
```

C++

```
__property LoginPrompt;
```

Description

Set LoginPrompt to true to provide login support when establishing a connection. LoginPrompt controls two things: the occurrence of the OnLogin event, and the appearance of a default login dialog that prompts users for a name and password when you include DBLogDlg.hpp in your unit (C++) or add DBLogDlg to your uses clause (Delphi). When the OnLogin event occurs and when the default login dialog appears depend on the type of connection component:

For TDatabase, TIBDatabase, and TSQLConnection, the dialog appears after the BeforeConnect event and before the

AfterConnect event, unless you supply an OnLogin event handler. If there is an OnLogin event handler, that event occurs in place of the login dialog, and there is no need to include DBLogDlg.hpp in your unit (C++) or add DBLogDlg to your uses clause (Delphi). If correct values for the user name and password are not supplied in the dialog or by the OnLogin event handler, the connection fails. The OnLogin event does not fire unless LoginPrompt is set to true.

For TADOConnection components, the dialog appears after the OnWillConnect event and before the BeforeConnect event. If there is an OnLogin event handler, that event occurs after the login dialog. If you do not include DBLogDlg.hpp in your unit (C++) or add DBLogDlg to your uses clause (Delphi), the OnLogin event occurs but there is no default login dialog. If correct values for the user name and password are not supplied in the dialog or by the OnLogin event handler, the connection fails. The OnLogin event does not fire unless LoginPrompt is set to true.

For DataSnap SOAP connection components, the dialog appears after the OnGetUsername event and before the BeforeConnect, AfterConnect, and OnLogin events. If the user cancels from the login dialog, no attempt is made to open a connection.

When LoginPrompt is false, the application must supply user name and password values programmatically:

For TDatabase, the user name and password can be supplied as USER_NAME and PASSWORD parameters in the Params property.

For TADOConnection, the user name and password can be supplied as the ConnectionString property.

For TSQLConnection, the user name, password, and database can be supplied as UserName, Password, and Database parameters in the Params property, or provided as connection parameters associated with the connection name.

For DataSnap SOAP connection components, there is no built-in use for the user name and password supplied by the login dialog. (The UserName and Password properties of TWebConnection are unrelated).

Warning: Storing hard-coded user name and password entries as property values or in code for an OnLogin event handler can compromise server security.

C++ Examples:

```
//
//This example demostrates the use of ADO for database conectivity.
//Example assumes that a TDBGrid is placed on the form.
//
void __fastcall TForm2::FormCreate(TObject *Sender)
{
    /* Login details */
    String UserName = "db_user_name";
    String PassWord = "db_pass_word";
    String Server = "my.db.server";

    /* Connection String */
    String ConnString =
        "Provider=SQLOLEDB.1;Persist Security Info=False;";
    ConnString +=
        "User ID=%s;Password=%s;Data Source=%s;Use Procedure for Prepare=1;";
    ConnString +=
        "Auto Translate=True;Packet Size=4096;Use Encryption for Data=False;";
    ConnString +=
        "Tag with column collation when possible=False";

    /* SQL Query */
    String SQLStr = "SELECT * FROM customer WHERE customer_id = :AnId;";

    /* All ADO variables */
    TADOConnection* ADOConn;
    TADOQuery* ADOQuery;
    TDataSource* DataSrc;
```

```
    TParameter* Param;

    /* Create an ADO connection */
    ADOConn = new TADOConnection(this);

    /* Setup the connection string */
    ADOConn->ConnectionString = Format(ConnString,
        ARRAYOFCONST((UserName, PassWord, Server)));

    /* Disable login prompt */
    ADOConn->LoginPrompt = False;

    try
    {
        ADOConn->Connected = true;
    }
    catch (EADOError *e)
    {
        MessageDlg("Error while connecting", mtError,
                    TMsgDlgButtons() << mbOK, 0);
        return;
    }

    /* Create the query */
    ADOQuery = new TADOQuery(this);
    ADOQuery->Connection = ADOConn;
    ADOQuery->SQL->Add(SQLStr);

    /* Update the parameter that was parsed from the SQL query: AnId */
    Param = ADOQuery->Parameters->ParamByName("AnId");
    Param->DataType = ftInteger;
    Param->Value = 1;

    /* Set the query to Prepared - will improve performance */
    ADOQuery->Prepared = true;

    try
    {
        ADOQuery->Active = true;
    }
    catch (EADOError *e)
    {
        MessageDlg("Error while connecting", mtError,
                    TMsgDlgButtons() << mbOK, 0);
        return;
    }

    /* Create the data source */
    DataSrc = new TDataSource(this);
    DataSrc->DataSet = ADOQuery;
    DataSrc->Enabled = true;

    /* Finally initilalize the grid */
    DBGrid1->DataSource = DataSrc;
}
```

Delphi Examples:

```
{
This example demostrates the use of ADO for database conectivity.
Example assumes that a TDBGrid is placed on the form.
}
procedure TForm2.FormCreate(Sender: TObject);
const
  { Connection string }
  ConnString =
  'Provider=SQLOLEDB.1;Persist Security Info=False;' +
```

```
    'User ID=%s;Password=%s;Data Source=%s;Use Procedure for Prepare=1;' +
    'Auto Translate=True;Packet Size=4096;Use Encryption for Data=False;'+
    'Tag with column collation when possible=False';

    { SQL Query }
    SQLStr = 'SELECT * FROM customer WHERE customer_id = :AnId;';

    { User access }
    UserName = 'db_user_name';
    PassWord = 'db_pass_word';
    Server = 'my.db.server';

  var
    ADOConn  : TADOConnection;
    ADOQuery : TADOQuery;
    DataSrc  : TDataSource;
    Param    : TParameter;

  begin
    { Create an ADO connection }
    ADOConn := TADOConnection.Create(Self);
    { Setup the provider engine }

    { Setup the connection string }
    ADOConn.ConnectionString := Format(ConnString,
      [UserName, PassWord, Server]);

    { Disable login prompt }
    ADOConn.LoginPrompt := False;

    try
      ADOConn.Connected := True;
    except
      on e: EADOError do
      begin
        MessageDlg('Error while connecting', mtError,
                   [mbOK], 0);

        Exit;
      end;
    end;

    { Create the query }
    ADOQuery := TADOQuery.Create(Self);
    ADOQuery.Connection := ADOConn;
    ADOQuery.SQL.Add(SQLStr);

    { Update the parameter that was parsed from the SQL query: AnId }
    Param := ADOQuery.Parameters.ParamByName('AnId');
    Param.DataType := ftInteger;
    Param.Value := 1;

    { Set the query to Prepared - will improve performance }
    ADOQuery.Prepared := true;

    try
      ADOQuery.Active := True;
    except
      on e: EADOError do
      begin
        MessageDlg('Error while doing query', mtError,
                   [mbOK], 0);

        Exit;
      end;
    end;
```

```
    { Create the data source }
    DataSrc := TDataSource.Create(Self);
    DataSrc.DataSet := ADOQuery;
    DataSrc.Enabled := true;

    { Finally initilalize the grid }
    DBGrid1.DataSource := DataSrc;
  end;
```

See Also

OnLogin

BeforeConnect

AfterConnect

OnGetUsername

OnLogin

OnLogin

Controlling Server Login

5.1.2.5.13 TIBDataBase.Params Property

Specifies the database parameters to pass to the InterBase server.

Pascal

```
property Params: TStrings;
```

C++

```
__property TStrings Params;
```

Description

Use Params to specify the database parameters to pass to the InterBase server.

Database parameters are passed to the server as text in order to establish the connection. For example:

```
user_name=sysdba
    password=masterkey
    sql_role_name=finance
    lc_ctype=WIN1252
```

For more information on character sets, refer to 'Character Sets and Collation Orders' in the InterBase Language Reference.

For other information, refer to the InterBase API Guide.

5.1.2.5.14 TIBDataBase.SQLDialect Property

Sets or returns the SQL dialect used by the client.

Pascal

```
property SQLDialect: Integer;
```

C++

```
__property int SQLDialect;
```

Description

Use the SQLDialect property to set or return the SQL dialect used by the client. If the connection is active, the SQLDialect

property cannot be set to a value greater than the database SQL dialect. If the connection is inactive, then on connect an OnDialectDowngradeWarning (see page 200) event may be fired if the SQLDialect is greater than the database SQL dialect. In such a case, the SQLDialect property will be downgraded to match the database SQL dialect.

5.1.2.5.15 TIBDataBase.SQLObjectCount Property

Returns the number of SQL objects.

Pascal

```
property SQLObjectCount: Integer;
```

C++

```
__property int SQLObjectCount;
```

Description

Use the SQLObjectCount property to return the number of SQL objects in the database.

SQL objects are usually defined as InterBase datasets, IBSQL (see page 493), and Blobs.

5.1.2.5.16 TIBDataBase.SQLObjects Property

Returns an SQL object.

Pascal

```
property SQLObjects [Index: Integer]: TIBBase;
```

C++

```
__property TIBBase SQLObjects[int Index];
```

Description

Use the SQLObjects property to return an SQL object based on its numeric index.

SQL objects are usually defined as InterBase datasets, IBSQL (see page 493), and Blobs.

5.1.2.5.17 TIBDataBase.TraceFlags Property

Specifies the database operations to track with the SQL Monitor at runtime.

Pascal

```
property TraceFlags: TTraceFlags;
```

C++

```
__property TTraceFlags TraceFlags;
```

Description

Use TraceFlags to specify which database operations the SQL Monitor should track in an application at runtime. TraceFlags is only meaningful for the SQL Monitor, which is provided to enable performance tuning and SQL debugging when working with remote SQL database servers.

Note: Normally trace options are set from the SQL Monitor rather than setting TraceFlags in application code.

The value of a session component's TraceFlags property determines the initial settings of the TraceFlags property for database components associated with the session.

Because TraceFlags is a set property, an application can specify different combinations of flags to monitor different combinations of statements. For example, the following statement limits monitoring to database connections and SQL statement preparation:

```
TraceFlags := [tfConnect, tfQPrepare];
    TraceFlags = [tfConnect, tfQPrepare];
```

5.1.2.5.18 TIBDataBase.TransactionCount Property

Returns the number of transactions associated with the TIBDataBase (see page 176) component.

Pascal

```
property TransactionCount: Integer;
```

C++

```
__property int TransactionCount;
```

Description

Use TransactionCount to return how many transactions are currently associated with the InterBase database component.

5.1.2.5.19 TIBDataBase.Transactions Property

Specifies a transaction for the given index.

Pascal

```
property Transactions [Index: Integer]: TIBTransaction;
```

C++

```
__property TIBTransaction Transactions[int Index];
```

Description

Given an integer index, Transactions returns the transaction at the given index. This is used internally for broadcasting important messages to attached components.

5.1.2.6 TIBDataBase Events

The events of the TIBDataBase class are listed here.

Published Events

	Name	Description
	AfterConnect (see page 198)	Occurs after a connection is established.
	AfterDisconnect (see page 199)	Occurs after the connection closes.
	BeforeConnect (see page 199)	Occurs immediately before establishing a connection.
	BeforeDisconnect (see page 199)	Occurs immediately before the connection closes.
	OnDialectDowngradeWarning (see page 200)	Occurs after the SQL dialect of the client connection is downgraded.
	OnIdleTimer (see page 200)	Occurs after a database connection times out.
	OnLogin (see page 201)	Occurs when an application connects to a database.

5.1.2.6.1 TIBDataBase.AfterConnect Event

Occurs after a connection is established.

Pascal

```
property AfterConnect;
```

C++

```
__property AfterConnect;
```

Description

Write an AfterConnect event handler to take application-specific actions immediately after the connection component opens a connection to the remote source of database information.

See Also

BeforeConnect

Connected

Connecting to a Database Server

5.1.2.6.2 TIBDataBase.AfterDisconnect Event

Occurs after the connection closes.

Pascal

```
property AfterDisconnect;
```

C++

```
__property AfterDisconnect;
```

Description

Write an AfterDisconnect event handler to take application-specific actions after the connection component drops a connection.

See Also

BeforeDisconnect

Connected

Disconnecting from a Database Server

5.1.2.6.3 TIBDataBase.BeforeConnect Event

Occurs immediately before establishing a connection.

Pascal

```
property BeforeConnect;
```

C++

```
__property BeforeConnect;
```

Description

Write a BeforeConnect event handler to take application-specific actions before the connection component opens a connection to the remote source of database information.

See Also

AfterConnect

Connected

Connecting to a Database Server

5.1.2.6.4 TIBDataBase.BeforeDisconnect Event

Occurs immediately before the connection closes.

Pascal

```
property BeforeDisconnect;
```

C++

```
__property BeforeDisconnect;
```

Description

Write a BeforeDisconnect event handler to take application-specific actions before dropping a connection.

Delphi Examples:

```
{
This BeforeDisconnect event handler makes sure that the
client dataset has applied all its pending updates before
terminating the connection to the server.
}
procedure TForm1.RemoteServer1BeforeDisconnect(Sender: TObject);
begin
  if (ClientDataSet1.ChangeCount > 0) then
    ClientDataSet1.ApplyUpdates(-1);
end;
```

See Also

AfterDisconnect

Connected

Disconnecting from a Database Server

5.1.2.6.5 TIBDataBase.OnDialectDowngradeWarning Event

Occurs after the SQL dialect of the client connection is downgraded.

Pascal

```
property OnDialectDowngradeWarning: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnDialectDowngradeWarning;
```

Description

Write an OnDialectDowngradeWarning event handler to take specific actions when the SQL dialect is downgraded.

For example, if the SQL dialect for your application is set to 3 and then a connection is made to a dialect 1 database, then the SQL dialect is downgraded to 1 and a OnDialectDowngradeWarning event is fired.

5.1.2.6.6 TIBDataBase.OnIdleTimer Event

Occurs after a database connection times out.

Pascal

```
property OnIdleTimer: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnIdleTimer;
```

Description

Write an OnIdleTimer event handler to take specific actions when the connection times out in the time specified by IdleTimer (see page 191).

5.1.2.6.7 TIBDataBase.OnLogin Event

Occurs when an application connects to a database.

Pascal

```
property OnLogin: TIBDatabaseLoginEvent;
```

C++

```
__property TIBDatabaseLoginEvent OnLogin;
```

Description

Write an OnLogin event handler to take specific actions when an application attempts to connect to a database. By default, a database login is required. The current USER_NAME is read from the Params (see page 196) property, and a standard Login dialog box opens. The dialog prompts for a user name and password combination, and then uses the values entered by the user to set the USER_NAME and PASSWORD values in the Params (see page 196) property. These values are then passed to the remote server.

Applications that provide alternative OnLogin event handlers must set the USER_NAME and PASSWORD values in LoginParams parameter of the event handler.

Note: OnLogin does not occur unless the LoginPrompt (see page 192) property is set to true.

5.1.3 TIBSchema Class

Class Hierarchy

Pascal

```
TIBSchema = class(TObject);
```

C++

```
class TIBSchema : public TObject;
```

File

IBDatabase

Description

This is class IBDatabase.TIBSchema.

5.1.3.1 TIBSchema Members

The following tables list the members exposed by TIBSchema.

Public Methods

	Name	Description
	FreeNodes (see page 202)	This is FreeNodes, a member of class TIBSchema.
	Has_COMPUTED_BLR (see page 202)	This is Has_COMPUTED_BLR, a member of class TIBSchema.
	Has_DEFAULT_VALUE (see page 202)	This is Has_DEFAULT_VALUE, a member of class TIBSchema.

5.1.3.2 TIBSchema Methods

The methods of the TIBSchema class are listed here.

Public Methods

	Name	Description
	FreeNodes (see page 202)	This is FreeNodes, a member of class TIBSchema.
	Has_COMPUTED_BLR (see page 202)	This is Has_COMPUTED_BLR, a member of class TIBSchema.
	Has_DEFAULT_VALUE (see page 202)	This is Has_DEFAULT_VALUE, a member of class TIBSchema.

5.1.3.2.1 TIBSchema.FreeNodes Method

Pascal

```
procedure FreeNodes; virtual; abstract;
```

C++

```
virtual __fastcall FreeNodes() = 0;
```

Description

This is FreeNodes, a member of class TIBSchema.

5.1.3.2.2 TIBSchema.Has_COMPUTED_BLR Method

Pascal

```
function Has_COMPUTED_BLR(Relation: String; Field: String): Boolean; virtual; abstract;
```

C++

```
virtual __fastcall Boolean Has_COMPUTED_BLR(UnicodeString Relation, UnicodeString Field) = 0;
```

Description

This is Has_COMPUTED_BLR, a member of class TIBSchema.

5.1.3.2.3 TIBSchema.Has_DEFAULT_VALUE Method

Pascal

```
function Has_DEFAULT_VALUE(Relation: String; Field: String): Boolean; virtual; abstract;
```

C++

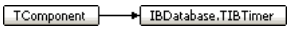
```
virtual __fastcall Boolean Has_DEFAULT_VALUE(UnicodeString Relation, UnicodeString Field) = 0;
```

Description

This is Has_DEFAULT_VALUE, a member of class TIBSchema.

5.1.4 TIBTimer Class

Class Hierarchy



Pascal

```
TIBTimer = class(TComponent);
```

C++

```
class TIBTimer : public TComponent;
```

File

IBDatabase

Description

This is class IBDatabase.TIBTimer.

5.1.4.1 TIBTimer Members

The following tables list the members exposed by TIBTimer.

Protected Methods

	Name	Description
	Timer (see page 205)	This is Timer, a member of class TIBTimer.

Public Methods

	Name	Description
	Create (see page 203)	Allocates memory and constructs a safely initialized instance of a component.
	Destroy (see page 204)	Disposes of the component and its owned components.

Published Events

	Name	Description
	OnTimer (see page 206)	This is OnTimer, a member of class TIBTimer.

Published Properties

	Name	Description
	Enabled (see page 205)	This is Enabled, a member of class TIBTimer.
	Interval (see page 206)	This is Interval, a member of class TIBTimer.

5.1.4.2 TIBTimer.Create Constructor

Allocates memory and constructs a safely initialized instance of a component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBTimer(TComponent * AOwner);
```

Description

All objects have a Create method that constructs the object. TComponent redefines Create so that, for components, Create also:

establishes the relationship of a component and its Owner, as indicated by the AOwner parameter

sets the ComponentStyle property to csInheritable, meaning that the component can be inherited by a descendant form type

It is not necessary to explicitly create components added in the form designer. These components are created automatically when the application is run, and they are destroyed when the application is closed.

For components created programmatically, that is, not created in the form designer, call Create and pass in an owner component as the AOwner parameter. The owner disposes of the component when it is destroyed. If the component is not owned, then use Free when it needs to be destroyed.

Tip: When passing in Self as the Owner parameter, consider what Self references. If a component creates another component in one of its methods, then Self refers to the first component and not the component being created, which is then owned by the first component.

Note: The TComponent constructor is virtual in part to allow polymorphic instantiation of class references. This is critical to the streaming system and to the form designer. Do not forget to use the override directive when declaring a new component's Create constructor.

See Also

Components

ComponentStyle

System::TObject::Create

Destroy

System::TObject::Free

Owner

5.1.4.3 TIBTimer.Destroy Destructor

Disposes of the component and its owned components.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBTimer();
```

Description

Do not call Destroy directly. Call Free instead. Free verifies that the component is not nil, and only then calls Destroy.

Never explicitly free a component in one of its own event handlers, nor free a component from the event handler of a component that it owns or contains.

To destroy a form, call its Release method. Release waits for all the form's event handlers and the event handlers of the form's components to finish executing before destroying the form.

Note: A form owns all the controls and non-visual components that are placed on it in design mode. When it is freed, all of these components are automatically freed as well. By default, all forms are owned by the global Application object. When an application terminates, it frees the global Application object, which frees all forms. For objects that are not components, and for components created with a nil owner, be sure to call Free after finishing with the object; otherwise the memory allocated for the object will be lost until the application terminates.

See Also

System::TObject::Destroy

DestroyComponents

Destroying

System::TObject::Free

Notification

Owner

Release

Components

5.1.4.4 TIBTimer Methods

The methods of the TIBTimer class are listed here.

Protected Methods

	Name	Description
	Timer (see page 205)	This is Timer, a member of class TIBTimer.

5.1.4.4.1 TIBTimer.Timer Method

Pascal

```
procedure Timer; dynamic;
```

C++

```
__fastcall Timer();
```

Description

This is Timer, a member of class TIBTimer.

5.1.4.5 TIBTimer Properties

The properties of the TIBTimer class are listed here.

Published Properties

	Name	Description
	Enabled (see page 205)	This is Enabled, a member of class TIBTimer.
	Interval (see page 206)	This is Interval, a member of class TIBTimer.

5.1.4.5.1 TIBTimer.Enabled Property

Pascal

```
property Enabled: Boolean;
```

C++

```
__property Boolean Enabled;
```

Description

This is Enabled, a member of class TIBTimer.

5.1.4.5.2 TIBTimer.Interval Property

Pascal

```
property Interval: Cardinal;
```

C++

```
__property unsigned Interval;
```

Description

This is Interval, a member of class TIBTimer.

5.1.4.6 TIBTimer Events

The events of the TIBTimer class are listed here.

Published Events

	Name	Description
	OnTimer (see page 206)	This is OnTimer, a member of class TIBTimer.

5.1.4.6.1 TIBTimer.OnTimer Event

Pascal

```
property OnTimer: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnTimer;
```

Description

This is OnTimer, a member of class TIBTimer.

5.1.5 TIBTransaction Class

TIBTransaction provides discrete transaction control over a one or more database connections in a database application.

Class Hierarchy

Pascal

```
TIBTransaction = class(TComponent);
```

C++

```
class TIBTransaction : public TComponent;
```

File

IBDatabase

Description

All TIBCustomDataSet descendants and TIBSQL need to use a transaction along with a database component to gain access to data in a database.

Note: In applications that connect an InterBaseExpress dataset to a client dataset, every query must be in its own transaction. You must use one transaction component for each query component.

See Also

Getting Started with InterBase Express

5.1.5.1 TIBTransaction Members

The following tables list the members exposed by TIBTransaction.

Protected Methods

	Name	Description
	Loaded (see page 212)	Initializes the component after the form file has been read into memory.
	Notification (see page 212)	Forwards notification messages to all owned components.
	SetHandle (see page 214)	This is SetHandle, a member of class TIBTransaction.

Public Methods

	Name	Description
	AddDatabase (see page 209)	Associates a database to the transaction.
	Call (see page 209)	Returns an error message based on the error code.
	CheckAutoStop (see page 210)	Performs the action that the AutoStopAction (see page 215) property specifies.
	CheckDatabasesInList (see page 210)	Checks for databases in the list.
	CheckInTransaction (see page 210)	Checks whether the transaction is active and whether there are any databases in the transaction's database list.
	CheckNotInTransaction (see page 210)	Checks that the transaction is not active and that there are no databases in the transaction's database list.
	Commit (see page 211)	Permanently stores updates, insertions, and deletions of data associated with the current transaction, and ends the current transactions.
	CommitRetaining (see page 211)	Commits the active transaction and retains the transaction context after a commit.
	Create (see page 208)	Creates an instance of a transaction component.
	Destroy (see page 208)	Destroys the instance of the transaction component.
	FindDatabase (see page 211)	Finds the index of the associated database.
	FindDefaultDatabase (see page 212)	Finds the default database for the transaction.
	RemoveDatabase (see page 213)	Disassociates a database from the transaction.
	RemoveDatabases (see page 213)	Disassociates all databases from the transaction.
	Rollback (see page 213)	Cancels all updates, insertions, and deletions for the current transaction and ends the transaction.
	RollbackRetaining (see page 214)	Cancels all updates, insertions, and deletions for the current transaction and retains the transaction context.
	StartTransaction (see page 214)	Begins a new transaction against the database server.

Public Properties

	Name	Description
	DatabaseCount (see page 216)	Indicates the number of databases that are part of the transaction.
	Databases (see page 216)	Returns the database at the given integer index.
	Handle (see page 217)	Returns the transaction handle.
	HandleIsShared (see page 217)	Indicates whether or not the handle is shared.
	InTransaction (see page 218)	Indicates whether a database transaction is in progress or not.
	SQLObjectCount (see page 218)	Returns the number of active datasets associated with the database component.
	SQLObjects (see page 218)	Provides an indexed array of all active datasets for a database component.
	TPB (see page 219)	Provides a read-only view of the transaction parameter buffer.
	TPBLength (see page 219)	Returns the length of the transaction parameter buffer.

Published Events

	Name	Description
	OnIdleTimer (see page 219)	Occurs after a transaction has timed out.

Published Properties

	Name	Description
	Active (see page 215)	Specifies whether or not a transaction is active.
	AutoStopAction (see page 215)	Specifies what action to take when automatically ending transactions.
	DefaultAction (see page 216)	Specifies what action a transaction should take upon timing out.
	DefaultDatabase (see page 217)	Sets or returns the default database for the transaction.
	IdleTimer (see page 217)	Specifies how long the transaction should wait before automatically committing or rolling back.
	Params (see page 218)	Returns the transaction parameter buffer associated with the transaction component.

5.1.5.2 TIBTransaction.Create Constructor

Creates an instance of a transaction component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBTransaction(TComponent * AOwner);
```

Description

Call (see page 209) Create to instantiate a transaction component at runtime. An application creates a transaction component in order to control the component's existence and set its properties and events.

Create instantiates a transaction component and creates an empty string list for the Params (see page 218) property.

5.1.5.3 TIBTransaction.Destroy Destructor

Destroys the instance of the transaction component.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBTransaction();
```

Description

Do not call Destroy directly in an application. Instead, call Free, which verifies that the transaction component is not already freed before calling Destroy.

Destroy disconnects from the database server, if necessary. It then frees the string resources allocated for the Params (see page 218) and SQLObjects (see page 218) properties before calling its inherited destructor.

5.1.5.4 TIBTransaction Methods

The methods of the TIBTransaction class are listed here.

Protected Methods

	Name	Description
	Loaded (see page 212)	Initializes the component after the form file has been read into memory.
	Notification (see page 212)	Forwards notification messages to all owned components.
	SetHandle (see page 214)	This is SetHandle, a member of class TIBTransaction.

Public Methods

	Name	Description
	AddDatabase (see page 209)	Associates a database to the transaction.
	Call (see page 209)	Returns an error message based on the error code.
	CheckAutoStop (see page 210)	Performs the action that the AutoStopAction (see page 215) property specifies.
	CheckDatabasesInList (see page 210)	Checks for databases in the list.
	CheckInTransaction (see page 210)	Checks whether the transaction is active and whether there are any databases in the transaction's database list.
	CheckNotInTransaction (see page 210)	Checks that the transaction is not active and that there are no databases in the transaction's database list.
	Commit (see page 211)	Permanently stores updates, insertions, and deletions of data associated with the current transaction, and ends the current transactions.
	CommitRetaining (see page 211)	Commits the active transaction and retains the transaction context after a commit.
	FindDatabase (see page 211)	Finds the index of the associated database.
	FindDefaultDatabase (see page 212)	Finds the default database for the transaction.
	RemoveDatabase (see page 213)	Disassociates a database from the transaction.
	RemoveDatabases (see page 213)	Disassociates all databases from the transaction.
	Rollback (see page 213)	Cancels all updates, insertions, and deletions for the current transaction and ends the transaction.
	RollbackRetaining (see page 214)	Cancels all updates, insertions, and deletions for the current transaction and retains the transaction context.
	StartTransaction (see page 214)	Begins a new transaction against the database server.

5.1.5.4.1 TIBTransaction.AddDatabase Method

Associates a database to the transaction.

Pascal

```
function AddDatabase(db: TIBDatabase): Integer;
```

C++

```
__fastcall int AddDatabase(TIBDatabase db);
```

Description

Call (see page 209) AddDatabase to associate a database to the transaction.

5.1.5.4.2 TIBTransaction.Call Method

Returns an error message based on the error code.

Pascal

```
function Call(ErrCode: ISC_STATUS; RaiseError: Boolean): ISC_STATUS;
```

C++

```
__fastcall ISC_STATUS Call(ISC_STATUS ErrCode, Boolean RaiseError);
```

Description

Call is an internal method used to make calls to the InterBase API, and gives you the option of raising an exception or returning an error based on the value of RaiseError.

5.1.5.4.3 TIBTransaction.CheckAutoStop Method

Performs the action that the AutoStopAction (see page 215) property specifies.

Pascal

```
procedure CheckAutoStop;
```

C++

```
__fastcall CheckAutoStop();
```

Description

Datasets call CheckAutoStop at times when it is appropriate to implicitly end a transaction. CheckAutoStop then performs the action that the AutoStopAction (see page 215) property specifies.

5.1.5.4.4 TIBTransaction.CheckDatabasesInList Method

Checks for databases in the list.

Pascal

```
procedure CheckDatabasesInList;
```

C++

```
__fastcall CheckDatabasesInList();
```

Description

Call (see page 209) CheckDatabasesInList to check if there are any databases in the list. If there are no databases in the list, an exception is raised.

5.1.5.4.5 TIBTransaction.CheckInTransaction Method

Checks whether the transaction is active and whether there are any databases in the transaction's database list.

Pascal

```
procedure CheckInTransaction;
```

C++

```
__fastcall CheckInTransaction();
```

Description

Call (see page 209) CheckInTransaction to determine whether the transaction is active and whether there are any databases in the transaction's database list. If either condition is false, an exception is raised.

5.1.5.4.6 TIBTransaction.CheckNotInTransaction Method

Checks that the transaction is not active and that there are no databases in the transaction's database list.

Pascal

```
procedure CheckNotInTransaction;
```

C++

```
__fastcall CheckNotInTransaction();
```

Description

Call (see page 209) CheckInTransaction (see page 210) to determine that the transaction is not active and that there are no databases in the transaction's database list. If either condition is false, an exception is raised.

5.1.5.4.7 TIBTransaction.Commit Method

Permanently stores updates, insertions, and deletions of data associated with the current transaction, and ends the current transactions.

Pascal

```
procedure Commit;
```

C++

```
__fastcall Commit();
```

Description

Call (see page 209) Commit to permanently store to the database server all updates, insertions, and deletions of data associated with the current transaction and then end the transaction. The current transaction is the last transaction started by calling StartTransaction (see page 214).

Note: Before calling Commit, an application may check the status of the InTransaction (see page 218) property. If an application calls Commit and there is no current transaction, an exception is raised.

5.1.5.4.8 TIBTransaction.CommitRetaining Method

Commits the active transaction and retains the transaction context after a commit.

Pascal

```
procedure CommitRetaining;
```

C++

```
__fastcall CommitRetaining();
```

Description

Call (see page 209) CommitRetaining to permanently store to the database server all updates, insertions, and deletions of data associated with the current transaction and then retain the transaction context. The current transaction is the last transaction started by calling StartTransaction (see page 214).

Note: Before calling CommitRetaining, an application may check the status of the InTransaction (see page 218) property. If an application calls CommitRetaining and there is no current transaction, an exception is raised.

5.1.5.4.9 TIBTransaction.FindDatabase Method

Finds the index of the associated database.

Pascal

```
function FindDatabase(db: TIBDatabase): Integer;
```

C++

```
__fastcall int FindDatabase(TIBDatabase db);
```

Description

Call (see page 209) FindDatabase to find the index of the associated database.

5.1.5.4.10 TIBTransaction.FindDefaultDatabase Method

Finds the default database for the transaction.

Pascal

```
function FindDefaultDatabase: TIBDatabase;
```

C++

```
__fastcall TIBDatabase FindDefaultDatabase();
```

Description

Call (see page 209) FindDefaultDatabase to find the default database associated with the transaction.

5.1.5.4.11 TIBTransaction.Loaded Method

Initializes the component after the form file has been read into memory.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

Do not call the protected Loaded method. The streaming system calls this method after it loads the component's form from a stream.

When the streaming system loads a form or data module from its form file, it first constructs the form component by calling its constructor, then reads its property values from the form file. After reading all the property values for all the components, the streaming system calls the Loaded methods of each component in the order the components were created. This gives the components a chance to initialize any data that depends on the values of other components or other parts of itself.

Note: All references to sibling components are resolved by the time Loaded is called. Loaded is the first place that sibling pointers can be used after being streamed in.

As implemented in TComponent, Loaded clears the csLoading flag in the ComponentState property, indicating that the component is no longer loading.

Warning: Loaded may be called multiple times on inherited forms. It is called every time a level of inheritance is streamed in. Do not allocate memory in an overridden Loaded method without first checking that the memory has not been allocated in a previous call.

See Also

ComponentState

Create

TReader

5.1.5.4.12 TIBTransaction.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

5.1.5.4.13 TIBTransaction.RemoveDatabase Method

Disassociates a database from the transaction.

Pascal

```
procedure RemoveDatabase(Idx: Integer);
```

C++

```
__fastcall RemoveDatabase(int Idx);
```

Description

Call (see page 209) RemoveDatabase to disassociate a specified database from the transaction.

5.1.5.4.14 TIBTransaction.RemoveDatabases Method

Disassociates all databases from the transaction.

Pascal

```
procedure RemoveDatabases;
```

C++

```
__fastcall RemoveDatabases();
```

Description

Call (see page 209) RemoveDatabases to disassociate all databases from the transaction.

5.1.5.4.15 TIBTransaction.Rollback Method

Cancels all updates, insertions, and deletions for the current transaction and ends the transaction.

Pascal

```
procedure Rollback;
```

C++

```
__fastcall Rollback();
```

Description

Call (see page 209) Rollback to cancel all updates, insertions, and deletions for the current transaction and to end the transaction. The current transaction is the last transaction started by calling StartTransaction (see page 214).

Note: Before calling Rollback, an application may check the status of the InTransaction (see page 218) property. If an application calls Rollback and there is no current transaction, an exception is raised.

5.1.5.4.16 TIBTransaction.RollbackRetaining Method

Cancels all updates, insertions, and deletions for the current transaction and retains the transaction context.

Pascal

```
procedure RollbackRetaining;
```

C++

```
__fastcall RollbackRetaining();
```

Description

Call (see page 209) RollbackRetaining to roll back to the database server all updates, insertions, and deletions of data associated with the current transaction and then retain the transaction context. The current transaction is the last transaction started by calling StartTransaction (see page 214).

Note: Before calling RollbackRetaining, an application may check the status of the InTransaction (see page 218) property. If an application calls RollbackRetaining and there is no current transaction, an exception is raised.

Note: You must install InterBase 6 to use this feature.

5.1.5.4.17 TIBTransaction.SetHandle Method

Pascal

```
procedure SetHandle(Value: TISC_TR_HANDLE);
```

C++

```
__fastcall SetHandle(TISC_TR_HANDLE Value);
```

Description

This is SetHandle, a member of class TIBTransaction.

5.1.5.4.18 TIBTransaction.StartTransaction Method

Begins a new transaction against the database server.

Pascal

```
procedure StartTransaction;
```

C++

```
__fastcall StartTransaction();
```

Description

Call (see page 209) StartTransaction to begin a new transaction against the database server. Before calling StartTransaction, an application should check the status of the InTransaction (see page 218) property. If InTransaction (see page 218) is true, indicating that a transaction is already in progress, a subsequent call to StartTransaction without first calling Commit (see page 211) or Rollback (see page 213) to end the current transaction raises an exception.

Updates, insertions, and deletions that take place after a call to StartTransaction are held by the server until an application calls Commit (see page 211) to save the changes or Rollback (see page 213) is to cancel them.

5.1.5.5 TIBTransaction Properties

The properties of the TIBTransaction class are listed here.

Public Properties

	Name	Description
	DatabaseCount (see page 216)	Indicates the number of databases that are part of the transaction.
	Databases (see page 216)	Returns the database at the given integer index.
	Handle (see page 217)	Returns the transaction handle.
	HandleIsShared (see page 217)	Indicates whether or not the handle is shared.
	InTransaction (see page 218)	Indicates whether a database transaction is in progress or not.
	SQLObjectCount (see page 218)	Returns the number of active datasets associated with the database component.
	SQLObjects (see page 218)	Provides an indexed array of all active datasets for a database component.
	TPB (see page 219)	Provides a read-only view of the transaction parameter buffer.
	TPBLength (see page 219)	Returns the length of the transaction parameter buffer.

Published Properties

	Name	Description
	Active (see page 215)	Specifies whether or not a transaction is active.
	AutoStopAction (see page 215)	Specifies what action to take when automatically ending transactions.
	DefaultAction (see page 216)	Specifies what action a transaction should take upon timing out.
	DefaultDatabase (see page 217)	Sets or returns the default database for the transaction.
	IdleTimer (see page 217)	Specifies how long the transaction should wait before automatically committing or rolling back.
	Params (see page 218)	Returns the transaction parameter buffer associated with the transaction component.

5.1.5.5.1 TIBTransaction.Active Property

Specifies whether or not a transaction is active.

Pascal

```
property Active: Boolean;
```

C++

```
__property Boolean Active;
```

Description

Use Active to determine or set a transaction's active state.

5.1.5.5.2 TIBTransaction.AutoStopAction Property

Specifies what action to take when automatically ending transactions.

Pascal

```
property AutoStopAction: TAutoStopAction;
```

C++

```
__property TAutoStopAction AutoStopAction;
```

Description

Use AutoStopAction to specify how the transaction should end when it is implicitly stopped. Transactions are implicitly stopped when AutoStopAction is not saNone and the last dataset associated with the transaction closes.

Note: This property only works with the closing of TDataSet descendants. If the transaction also includes TIBSQL components, AutoStopAction should be saNone, because the transaction can't tell whether there are any active TIBSQL components.

5.1.5.5.3 TIBTransaction.DatabaseCount Property

Indicates the number of databases that are part of the transaction.

Pascal

```
property DatabaseCount: Integer;
```

C++

```
__property int DatabaseCount;
```

Description

Use DatabaseCount to determine the number of databases involved in a transaction.

5.1.5.5.4 TIBTransaction.Databases Property

Returns the database at the given integer index.

Pascal

```
property Databases [Index: Integer]: TIBDatabase;
```

C++

```
__property TIBDatabase Databases[int Index];
```

Description

Use Databases to return the database at the given integer index.

5.1.5.5.5 TIBTransaction.DefaultAction Property

Specifies what action a transaction should take upon timing out.

Pascal

```
property DefaultAction: TTransactionAction;
```

C++

```
__property TTransactionAction DefaultAction;
```

Description

Use DefaultAction to what action the transaction should take when the IdleTimer (see page 217) limit is met.

5.1.5.5.6 TIBTransaction.DefaultDatabase Property

Sets or returns the default database for the transaction.

Pascal

```
property DefaultDatabase: TIBDatabase;
```

C++

```
__property TIBDatabase DefaultDatabase;
```

Description

Use DefaultDatabase to set or return the default database for the transaction.

5.1.5.5.7 TIBTransaction.Handle Property

Returns the transaction handle.

Pascal

```
property Handle: TISC_TR_HANDLE;
```

C++

```
__property TISC_TR_HANDLE Handle;
```

Description

Use Handle to retrieve a handle to the transaction. Handle is assigned an initial value when a transaction is started.

5.1.5.5.8 TIBTransaction.HandleIsShared Property

Indicates whether or not the handle is shared.

Pascal

```
property HandleIsShared: Boolean;
```

C++

```
__property Boolean HandleIsShared;
```

Description

HandleIsShared returns true when the transaction handle is shared by more than one transaction component.

5.1.5.5.9 TIBTransaction.IdleTimer Property

Specifies how long the transaction should wait before automatically committing or rolling back.

Pascal

```
property IdleTimer: Integer;
```

C++

```
__property int IdleTimer;
```

Description

Use IdleTimer to indicate how long a transaction should be allowed to remain idle before automatically committing or rolling back the data. Use DefaultAction (see page 216) to determine which action the transaction should take.

5.1.5.5.10 TIBTransaction.InTransaction Property

Indicates whether a database transaction is in progress or not.

Pascal

```
property InTransaction: Boolean;
```

C++

```
__property Boolean InTransaction;
```

Description

Examine InTransaction at run-time to determine if a database transaction is currently in progress. InTransaction is true if a transaction is in progress, false otherwise.

The value of InTransaction cannot be changed directly. Calling StartTransaction (see page 214) sets InTransaction to true. Calling Commit (see page 211) or Rollback (see page 213) sets InTransaction to false.

5.1.5.5.11 TIBTransaction.Params Property

Returns the transaction parameter buffer associated with the transaction component.

Pascal

```
property Params: TStrings;
```

C++

```
__property TStrings Params;
```

Description

Use Params to examine and set parameters in the transaction parameter buffer. Refer to the Interbase API Guide for the names of the parameters to provide.

5.1.5.5.12 TIBTransaction.SQLObjectCount Property

Returns the number of active datasets associated with the database component.

Pascal

```
property SQLObjectCount: Integer;
```

C++

```
__property int SQLObjectCount;
```

Description

Use the SQLObjectCount property to return the number currently active InterBase datasets, TIBSQL objects, and Blobs associated with the database component. As SQL objects are opened and closed, this value changes appropriately.

5.1.5.5.13 TIBTransaction.SQLObjects Property

Provides an indexed array of all active datasets for a database component.

Pascal

```
property SQLObjects [Index: Integer]: TIBBase;
```

C++

```
__property TIBBase SQLObjects[int Index];
```

Description

Use the SQLObjects to access active InterBase datasets, TIBSQL objects, and Blobs associated with the database component.

5.1.5.5.14 TIBTransaction.TPB Property

Provides a read-only view of the transaction parameter buffer.

Pascal

```
property TPB: PChar;
```

C++

```
__property const char * TPB;
```

Description

Use TPB view the transaction parameter buffer. To write to the transaction parameter buffer, use the Params (see page 218) property.

5.1.5.5.15 TIBTransaction.TPBLength Property

Returns the length of the transaction parameter buffer.

Pascal

```
property TPBLength: Short;
```

C++

```
__property Short TPBLength;
```

Description

Use TPBLength to retrieve the length of the transaction parameter buffer

5.1.5.6 TIBTransaction Events

The events of the TIBTransaction class are listed here.

Published Events

	Name	Description
	OnIdleTimer (see page 219)	Occurs after a transaction has timed out.

5.1.5.6.1 TIBTransaction.OnIdleTimer Event

Occurs after a transaction has timed out.

Pascal

```
property OnIdleTimer: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnIdleTimer;
```

Description

Write an OnIdleTimer event handler to take specific actions after a transaction is allowed to remain idle for the number of seconds specified by IdleTimer (see page 217).

5.2 Interfaces

The following table lists interfaces in this documentation.

Interfaces

	Name	Description
	IIBEventNotifier (see page 221)	IIBEventNotifier allows a TIBDataBase (see page 176) component to automatically register interest in events on the InterBase server.

5.2.1 IIBEventNotifier Interface

IIBEventNotifier allows a TIBDataBase (see page 176) component to automatically register interest in events on the InterBase server.

Class Hierarchy

IBDatabase.IIBEventNotifier

Pascal

```
IIBEventNotifier = interface;
```

C++

```
__interface IIBEventNotifier;
```

File

IBDatabase

Description

TIBDatabase uses the IIBEventNotifier interface to communicate with objects such as TIBEvents that register an interest in events on the InterBase server.

Note: VCL Method declarations that use IIBEventNotifier use the _di_IIBEventNotifier type instead. This is a DelphiInterface wrapper around the IIBEventNotifier interface:

```
typedef System::DelphiInterface< IIBEventNotifier > _di_IIBEventNotifier;
```

5.2.1.1 IIBEventNotifier Members

The following tables list the members exposed by IIBEventNotifier.

Public Methods

	Name	Description
	GetAutoRegister (see page 222)	Indicates whether the database should automatically register events after it connects to the server.
	RegisterEvents (see page 222)	Registers interest in a set of events with the InterBase Event manager.
	UnRegisterEvents (see page 222)	Unregisters interest in the events previously registered by a call to RegisterEvents (see page 222).

5.2.1.2 IIBEventNotifier Methods

The methods of the IIBEventNotifier class are listed here.

Public Methods

	Name	Description
	GetAutoRegister (see page 222)	Indicates whether the database should automatically register events after it connects to the server.
	RegisterEvents (see page 222)	Registers interest in a set of events with the InterBase Event manager.
	UnRegisterEvents (see page 222)	Unregisters interest in the events previously registered by a call to RegisterEvents (see page 222).

5.2.1.2.1 IIBEventNotifier.GetAutoRegister Method

Indicates whether the database should automatically register events after it connects to the server.

Pascal

```
function GetAutoRegister: Boolean;
```

C++

```
__fastcall Boolean GetAutoRegister();
```

Description

When TIBDataBase (see page 176) connects to an InterBase server, it calls the GetAutoRegister method for every object in its list of event notifiers. For any event notifier that returns true, it calls the RegisterEvents (see page 222) method, so that the event notifier can register interest in its events with the InterBase Event Manager.

5.2.1.2.2 IIBEventNotifier.RegisterEvents Method

Registers interest in a set of events with the InterBase Event manager.

Pascal

```
procedure RegisterEvents;
```

C++

```
__fastcall RegisterEvents();
```

Description

RegisterEvents tells the event notifier to register its interest in a set of events with the InterBase Event manager.

5.2.1.2.3 IIBEventNotifier.UnRegisterEvents Method

Unregisters interest in the events previously registered by a call to RegisterEvents (see page 222).

Pascal

```
procedure UnRegisterEvents;
```

C++

```
__fastcall UnRegisterEvents();
```

Description

UnRegisterEvents cancels the interest in the events that was previously registered by a call to RegisterEvents (see page 222). TIBDataBase (see page 176) uses this interface to unregister all of its event notifiers when it disconnects from the InterBase

server.

5.3 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	GenerateDPB (see page 224)	This is function IBDatabase.GenerateDPB.
	GenerateTPB (see page 224)	This is function IBDatabase.GenerateTPB.

5.3.1 IBDatabase.GenerateDPB Function

Pascal

```
procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);
```

C++

```
GenerateDPB(TStrings sl, UnicodeString DPB, Short DPBLength);
```

File

IBDatabase

Description

This is function IBDatabase.GenerateDPB.

5.3.2 IBDatabase.GenerateTPB Function

Pascal

```
procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
```

C++

```
GenerateTPB(TStrings sl, UnicodeString TPB, Short TPBLength);
```

File

IBDatabase

Description

This is function IBDatabase.GenerateTPB.

5.4 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TAutoStopAction (see page 225)	TAutoStopAction indicates the action to take on automatically ended transactions.
	TTransactionAction (see page 226)	This is record IBDatabase.TTransactionAction.

5.4.1 IBDatabase.TAutoStopAction Enumeration

TAutoStopAction indicates the action to take on automatically ended transactions.

Pascal

```
TAutoStopAction = (
  saNone,
  saRollback,
  saCommit,
  saRollbackRetaining,
  saCommitRetaining
);
```

C++

```
enum TAutoStopAction {
  saNone,
  saRollback,
  saCommit,
  saRollbackRetaining,
  saCommitRetaining
};
```

File

IBDatabase

Description

TAutoStopAction values are:

Value Meaning

saNone Transactions are not implicitly stopped.

saRollback Transactions are rolled back and closed when stopped implicitly.

saCommit Transactions are committed and closed when stopped implicitly.

saRollbackRetaining Transactions are not stopped when the last dataset closes, but all updates, insertions, and deletions of data associated with the current transaction are backed out. This option is only supported in InterBase 6.0 and later.

saCommitRetaining Transactions are not stopped when the last dataset closes, but all updates, insertions, and deletions of data associated with the current transaction are committed.

5.4.2 IBDatabase.TTransactionAction Enumeration

Pascal

```
  TTransactionAction = (
    TARollback,
    TACommit,
    TARollbackRetaining,
    TACommitRetaining
  );
```

C++

```
  enum TTransactionAction {
    TARollback,
    TACommit,
    TARollbackRetaining,
    TACommitRetaining
  };
```

File

IBDatabase

Description

This is record IBDatabase.TTransactionAction.

5.5 Types

The following table lists types in this documentation.

Types

Name	Description
TIBDatabaseLoginEvent (see page 227)	TIBDatabaseLoginEvent is the type of event handlers that respond when an IB (see page 1) database component forms a connection to an InterBase server.
TIBFileName (see page 227)	This is type IBDatabase.TIBFileName.

5.5.1 IBDatabase.TIBDatabaseLoginEvent Type

TIBDatabaseLoginEvent is the type of event handlers that respond when an IB (see page 1) database component forms a connection to an InterBase server.

Pascal

```
TIBDatabaseLoginEvent = procedure (Database: TIBDatabase; LoginParams: TStrings) of object;
```

C++

```
(Database: TIBDatabase; LoginParams: TStrings) ( TIBDatabaseLoginEvent)();
```

File

IBDatabase

Description

Database is the connection component that establishes the connection to the InterBase server.

LoginParams is a temporary string list that contains the login parameters for forming the connection. The event handler can examine or change any of these value. Each parameter is of the form Name=Value. Of particular importance are the parameters with the names USER_NAME and PASSWORD. This string list does not persist: the IB (see page 1) database component frees it when it is no longer needed.

See Also

Procedural types

Creating events

5.5.2 IBDatabase.TIBFileName Type

Pascal

```
TIBFileName = type string;
```

C++

```
type string TIBFileName;
```

File

IBDatabase

Description

This is type IBDatabase.TIBFileName.

5.6 Constants

The following table lists constants in this documentation.

Constants

Name	Description
DPBConstantNames (see page 229)	This is constant IBDatabase.DPBConstantNames.
DPBPrefix (see page 230)	This is constant IBDatabase.DPBPrefix.
TPBConstantNames (see page 230)	This is constant IBDatabase.TPBConstantNames.
TPBPrefix (see page 230)	This is constant IBDatabase.TPBPrefix.

5.6.1 IBDatabase.DPBConstantNames Constant

Pascal

```
DPBConstantNames: array[1..isc_dpb_last_dpb_constant] of string = ( 'cdd_pathname',
'allocation', 'journal', 'page_size', 'num_buffers', 'buffer_length', 'debug',
'garbage_collect', 'verify', 'sweep', 'enable_journal', 'disable_journal', 'dbkey_scope',
'number_of_users', 'trace', 'no_garbage_collect', 'damaged', 'license', 'sys_user_name',
'encrypt_key', 'activate_shadow', 'sweep_interval', 'delete_shadow', 'force_write',
'begin_log', 'quit_log', 'no_reserve', 'user_name', 'password', 'password_enc',
'sys_user_name_enc', 'interp', 'online_dump', 'old_file_size', 'old_num_files', 'old_file',
'old_start_page', 'old_start_seqno', 'old_start_file', 'drop_walfile', 'old_dump_id',
'wal_backup_dir', 'wal_chkptlen', 'wal_numbufs', 'wal_bufsize', 'wal_grp_cmt_wait',
'lc_messages', 'lc_ctype', 'cache_manager', 'shutdown', 'online', 'shutdown_delay',
'reserved', 'overwrite', 'sec_attach', 'disable_wal', 'connect_timeout',
'dummy_packet_interval', 'gbak_attach', 'sql_role_name', 'set_page_buffers',
'working_directory', 'sql_dialect', 'set_db_readonly', 'set_db_sql_dialect', 'gfix_attach',
'gstat_attach' );
```

C++

```
array[1..isc_dpb_last_dpb_constant] of string DPBConstantNames = ( 'cdd_pathname',
'allocation', 'journal', 'page_size', 'num_buffers', 'buffer_length', 'debug',
'garbage_collect', 'verify', 'sweep', 'enable_journal', 'disable_journal', 'dbkey_scope',
'number_of_users', 'trace', 'no_garbage_collect', 'damaged', 'license', 'sys_user_name',
'encrypt_key', 'activate_shadow', 'sweep_interval', 'delete_shadow', 'force_write',
'begin_log', 'quit_log', 'no_reserve', 'user_name', 'password', 'password_enc',
'sys_user_name_enc', 'interp', 'online_dump', 'old_file_size', 'old_num_files', 'old_file',
'old_start_page', 'old_start_seqno', 'old_start_file', 'drop_walfile', 'old_dump_id',
'wal_backup_dir', 'wal_chkptlen', 'wal_numbufs', 'wal_bufsize', 'wal_grp_cmt_wait',
'lc_messages', 'lc_ctype', 'cache_manager', 'shutdown', 'online', 'shutdown_delay',
'reserved', 'overwrite', 'sec_attach', 'disable_wal', 'connect_timeout',
'dummy_packet_interval', 'gbak_attach', 'sql_role_name', 'set_page_buffers',
'working_directory', 'sql_dialect', 'set_db_readonly', 'set_db_sql_dialect', 'gfix_attach',
'gstat_attach' );
```

File

IBDatabase

Description

This is constant IBDatabase.DPBConstantNames.

5.6.2 IBDatabase.DPBPrefix Constant

Pascal

```
DPBPrefix = 'isc_dpb_';
```

C++

```
DPBPrefix = 'isc_dpb_';
```

File

IBDatabase

Description

This is constant IBDatabase.DPBPrefix.

5.6.3 IBDatabase.TPBConstantNames Constant

Pascal

```
TPBConstantNames: array[1..isc_tpb_last_tpb_constant] of string = ( 'consistency',
'concurrency', 'shared', 'protected', 'exclusive', 'wait', 'nowait', 'read', 'write',
'lock_read', 'lock_write', 'verb_time', 'commit_time', 'ignore_limbo', 'read_committed',
'autocommit', 'rec_version', 'no_rec_version', 'restart_requests', 'no_auto_undo' );
```

C++

```
array[1..isc_tpb_last_tpb_constant] of string TPBConstantNames = ( 'consistency',
'concurrency', 'shared', 'protected', 'exclusive', 'wait', 'nowait', 'read', 'write',
'lock_read', 'lock_write', 'verb_time', 'commit_time', 'ignore_limbo', 'read_committed',
'autocommit', 'rec_version', 'no_rec_version', 'restart_requests', 'no_auto_undo' );
```

File

IBDatabase

Description

This is constant IBDatabase.TPBConstantNames.

5.6.4 IBDatabase.TPBPrefix Constant

Pascal

```
TPBPrefix = 'isc_tpb_';
```

C++

```
TPBPrefix = 'isc_tpb_';
```

File

IBDatabase

Description

This is constant IBDatabase.TPBPrefix.

6 IBDatabaseInfo Namespace

This is namespace IBDatabaseInfo.

Topics

Name	Description
Classes (see page 232)	The following table lists classes in this documentation.

Classes

	Name	Description
	TIBDatabaseInfo (see page 232)	TIBDatabaseInfo returns information about the attached database.

6.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBDatabaseInfo (see page 232)	TIBDatabaseInfo returns information about the attached database.

6.1.1 TIBDatabaseInfo Class

TIBDatabaseInfo returns information about the attached database.

Class Hierarchy

Pascal

```
TIBDatabaseInfo = class(TComponent);
```

C++

```
class TIBDatabaseInfo : public TComponent;
```

File

IBDatabaseInfo

Description

Use TIBDatabaseInfo to fetch information about the attached database, such as the version of the online disk structure (ODS) used by the attachment, the number of database cache buffers allocated, the number of database pages read from or written to, or write-ahead log information.

See Also

Getting Started with InterBase Express

6.1.1.1 TIBDatabaseInfo Members

The following tables list the members exposed by TIBDatabaseInfo.

Protected Fields

	Name	Description
	FBackoutCount (see page 235)	This is FBackoutCount, a member of class TIBDatabaseInfo.
	FDatabase (see page 235)	This is FDatabase, a member of class TIBDatabaseInfo.
	FDeleteCount (see page 236)	This is FDeleteCount, a member of class TIBDatabaseInfo.
	FExpungeCount (see page 236)	This is FExpungeCount, a member of class TIBDatabaseInfo.
	FIBLoaded (see page 236)	This is FIBLoaded, a member of class TIBDatabaseInfo.
	FInsertCount (see page 236)	This is FInsertCount, a member of class TIBDatabaseInfo.
	FPurgeCount (see page 236)	This is FPurgeCount, a member of class TIBDatabaseInfo.
	FReadIdxCount (see page 237)	This is FReadIdxCount, a member of class TIBDatabaseInfo.

	FReadSeqCount (see page 237)	This is FReadSeqCount, a member of class TIBDatabaseInfo.
	FUpdateCount (see page 237)	This is FUpdateCount, a member of class TIBDatabaseInfo.
	FUserNames (see page 237)	This is FUserNames, a member of class TIBDatabaseInfo.

Protected Methods

	Name	Description
	GetAllocation (see page 239)	This is GetAllocation, a member of class TIBDatabaseInfo.
	GetBackoutCount (see page 239)	This is GetBackoutCount, a member of class TIBDatabaseInfo.
	GetBaseLevel (see page 239)	This is GetBaseLevel, a member of class TIBDatabaseInfo.
	GetCurrentMemory (see page 239)	This is GetCurrentMemory, a member of class TIBDatabaseInfo.
	GetDBFileName (see page 239)	This is GetDBFileName, a member of class TIBDatabaseInfo.
	GetDBImplementationClass (see page 240)	This is GetDBImplementationClass, a member of class TIBDatabaseInfo.
	GetDBImplementationNo (see page 240)	This is GetDBImplementationNo, a member of class TIBDatabaseInfo.
	GetDBSiteName (see page 240)	This is GetDBSiteName, a member of class TIBDatabaseInfo.
	GetDBSQLDialect (see page 240)	This is GetDBSQLDialect, a member of class TIBDatabaseInfo.
	GetDeleteCount (see page 241)	This is GetDeleteCount, a member of class TIBDatabaseInfo.
	GetExpungeCount (see page 241)	This is GetExpungeCount, a member of class TIBDatabaseInfo.
	GetFetches (see page 241)	This is GetFetches, a member of class TIBDatabaseInfo.
	GetForcedWrites (see page 241)	This is GetForcedWrites, a member of class TIBDatabaseInfo.
	GetInsertCount (see page 241)	This is GetInsertCount, a member of class TIBDatabaseInfo.
	GetMarks (see page 242)	This is GetMarks, a member of class TIBDatabaseInfo.
	GetMaxMemory (see page 242)	This is GetMaxMemory, a member of class TIBDatabaseInfo.
	GetNoReserve (see page 242)	This is GetNoReserve, a member of class TIBDatabaseInfo.
	GetNumBuffers (see page 243)	This is GetNumBuffers, a member of class TIBDatabaseInfo.
	GetODSMajorVersion (see page 243)	This is GetODSMajorVersion, a member of class TIBDatabaseInfo.
	GetODSMinorVersion (see page 243)	This is GetODSMinorVersion, a member of class TIBDatabaseInfo.
	GetOperationCounts (see page 243)	This is GetOperationCounts, a member of class TIBDatabaseInfo.
	GetPageSize (see page 244)	This is GetPageSize, a member of class TIBDatabaseInfo.
	GetPurgeCount (see page 244)	This is GetPurgeCount, a member of class TIBDatabaseInfo.
	GetReadIdxCount (see page 244)	This is GetReadIdxCount, a member of class TIBDatabaseInfo.
	GetReadOnly (see page 244)	This is GetReadOnly, a member of class TIBDatabaseInfo.
	GetReads (see page 244)	This is GetReads, a member of class TIBDatabaseInfo.
	GetReadSeqCount (see page 245)	This is GetReadSeqCount, a member of class TIBDatabaseInfo.
	GetStringDatabaseInfo (see page 245)	This is GetStringDatabaseInfo, a member of class TIBDatabaseInfo.
	GetSweepInterval (see page 245)	This is GetSweepInterval, a member of class TIBDatabaseInfo.
	GetUpdateCount (see page 245)	This is GetUpdateCount, a member of class TIBDatabaseInfo.
	GetUserNames (see page 246)	This is GetUserNames, a member of class TIBDatabaseInfo.
	GetVersion (see page 246)	This is GetVersion, a member of class TIBDatabaseInfo.
	GetWrites (see page 246)	This is GetWrites, a member of class TIBDatabaseInfo.

Public Methods

	Name	Description
	Call (see page 238)	Returns an error message based on the error code.
V	Create (see page 234)	Creates an instance of a DatabaseInfo component.
V	Destroy (see page 234)	Destroys an instance of a DatabaseInfo component.
	GetLongDatabaseInfo (see page 242)	Returns a specified long integer that describes some aspect of the associated database.

Public Properties

	Name	Description
R	Allocation (see page 247)	Returns the number of database pages allocated.
R	BackoutCount (see page 247)	Returns the number of removals of a version of a record.
R	BaseLevel (see page 248)	Returns the database version number.
R	CurrentMemory (see page 248)	Returns the amount of server memory currently in use.
R	DBFileName (see page 248)	Returns the database filename.

	DBImplementationClass (see page 249)	Returns the database implementation class number.
	DBImplementationNo (see page 249)	Returns the database implementation number.
	DBSiteName (see page 249)	Returns the database site name.
	DBSQLDialect (see page 249)	Returns the SQL dialect.
	DeleteCount (see page 250)	Returns the number of database deletes since the database was last attached.
	ExpungeCount (see page 250)	Returns the number of removals of a record and all of its ancestors.
	Fetches (see page 250)	Returns the number of reads from the memory buffer cache.
	ForcedWrites (see page 250)	Returns the mode in which database writes are performed.
	InsertCount (see page 251)	Returns number of inserts into the database since the database was last attached.
	Marks (see page 251)	Returns the number of writes to the memory buffer cache.
	MaxMemory (see page 251)	Returns the maximum amount of memory used at one time since the first process attached to the database.
	NoReserve (see page 251)	Returns whether or not space is reserved on each database page for holding backup versions of modified records.
	NumBuffers (see page 252)	Returns the number of memory buffers currently allocated.
	ODSMajorVersion (see page 252)	Returns the on disk structure (ODS) major version number.
	ODSMinorVersion (see page 252)	Returns the on disk structure (ODS) minor version number.
	PageSize (see page 252)	Returns the number of bytes per page of the attached database.
	PurgeCount (see page 253)	Returns the number of removals of fully mature records from the database.
	ReadIdxCount (see page 253)	Returns the number of reads done via an index since the database was last attached.
	ReadOnly (see page 253)	Indicates whether or not the database is read only.
	Reads (see page 253)	Returns the number of page reads from the database.
	ReadSeqCount (see page 254)	Returns the number of sequential database reads done on each table since the database was last attached.
	SweepInterval (see page 254)	Returns the number of transactions that are committed between 'sweeps.'
	UpdateCount (see page 254)	Returns the number of database updates since the database was last attached.
	UserNames (see page 255)	Returns the names of all users currently attached to the database.
	Version (see page 255)	Returns the version of the database implementation.
	Writes (see page 255)	Returns the number of page writes to the database.

Published Properties

	Name	Description
	Database (see page 248)	Sets or returns the database.

6.1.1.2 TIBDatabaseInfo.Create Constructor

Creates an instance of a DatabaseInfo component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBDatabaseInfo(TComponent * AOwner);
```

Description

Call (see page 238) Create to instantiate a DatabaseInfo component declared in an application.

6.1.1.3 TIBDatabaseInfo.Destroy Destructor

Destroys an instance of a DatabaseInfo component.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBDatabaseInfo();
```

Description

Do not call Destroy directly. Instead call Free to verify that the DatabaseInfo component is not already freed before calling Destroy. Destroy disconnects from the server, frees the parameter list, and calls its inherited Destroy destructor.

6.1.1.4 TIBDatabaseInfo Fields

The fields of the TIBDatabaseInfo class are listed here.

Protected Fields

	Name	Description
	FBackoutCount (see page 235)	This is FBackoutCount, a member of class TIBDatabaseInfo.
	FDatabase (see page 235)	This is FDatabase, a member of class TIBDatabaseInfo.
	FDeleteCount (see page 236)	This is FDeleteCount, a member of class TIBDatabaseInfo.
	FExpungeCount (see page 236)	This is FExpungeCount, a member of class TIBDatabaseInfo.
	FIBLoaded (see page 236)	This is FIBLoaded, a member of class TIBDatabaseInfo.
	FInsertCount (see page 236)	This is FInsertCount, a member of class TIBDatabaseInfo.
	FPurgeCount (see page 236)	This is FPurgeCount, a member of class TIBDatabaseInfo.
	FReadIdxCount (see page 237)	This is FReadIdxCount, a member of class TIBDatabaseInfo.
	FReadSeqCount (see page 237)	This is FReadSeqCount, a member of class TIBDatabaseInfo.
	FUpdateCount (see page 237)	This is FUpdateCount, a member of class TIBDatabaseInfo.
	FUserNames (see page 237)	This is FUserNames, a member of class TIBDatabaseInfo.

6.1.1.4.1 TIBDatabaseInfo.FBackoutCount Field

Pascal

```
FBackoutCount: TStringList;
```

C++

```
TStringList FBackoutCount;
```

Description

This is FBackoutCount, a member of class TIBDatabaseInfo.

6.1.1.4.2 TIBDatabaseInfo.FDatabase Field

Pascal

```
FDatabase: TIBDatabase;
```

C++

```
TIBDatabase FDatabase;
```

Description

This is FDatabase, a member of class TIBDatabaseInfo.

6.1.1.4.3 TIBDatabaseInfo.FDeleteCount Field

Pascal

```
FDeleteCount: TStringList;
```

C++

```
TStringList FDeleteCount;
```

Description

This is FDeleteCount, a member of class TIBDatabaseInfo.

6.1.1.4.4 TIBDatabaseInfo.FExpungeCount Field

Pascal

```
FExpungeCount: TStringList;
```

C++

```
TStringList FExpungeCount;
```

Description

This is FExpungeCount, a member of class TIBDatabaseInfo.

6.1.1.4.5 TIBDatabaseInfo.FIBLoaded Field

Pascal

```
FIBLoaded: Boolean;
```

C++

```
Boolean FIBLoaded;
```

Description

This is FIBLoaded, a member of class TIBDatabaseInfo.

6.1.1.4.6 TIBDatabaseInfo.FInsertCount Field

Pascal

```
FInsertCount: TStringList;
```

C++

```
TStringList FInsertCount;
```

Description

This is FInsertCount, a member of class TIBDatabaseInfo.

6.1.1.4.7 TIBDatabaseInfo.FPurgeCount Field

Pascal

```
FPurgeCount: TStringList;
```

C++

```
TStringList FPurgeCount;
```

Description

This is FPurgeCount, a member of class TIBDatabaseInfo.

6.1.1.4.8 TIBDatabaseInfo.FReadIdxCount Field

Pascal

```
FReadIdxCount: TStringList;
```

C++

```
TStringList FReadIdxCount;
```

Description

This is FReadIdxCount, a member of class TIBDatabaseInfo.

6.1.1.4.9 TIBDatabaseInfo.FReadSeqCount Field

Pascal

```
FReadSeqCount: TStringList;
```

C++

```
TStringList FReadSeqCount;
```

Description

This is FReadSeqCount, a member of class TIBDatabaseInfo.

6.1.1.4.10 TIBDatabaseInfo.FUpdateCount Field

Pascal

```
FUpdateCount: TStringList;
```

C++

```
TStringList FUpdateCount;
```

Description

This is FUpdateCount, a member of class TIBDatabaseInfo.

6.1.1.4.11 TIBDatabaseInfo.FUserNames Field

Pascal

```
FUserNames: TStringList;
```

C++

```
TStringList FUserNames;
```

Description

This is FUserNames, a member of class TIBDatabaseInfo.

6.1.1.5 TIBDatabaseInfo Methods

The methods of the TIBDatabaseInfo class are listed here.

Protected Methods

	Name	Description
	GetAllocation (see page 239)	This is GetAllocation, a member of class TIBDatabaseInfo.
	GetBackoutCount (see page 239)	This is GetBackoutCount, a member of class TIBDatabaseInfo.
	GetBaseLevel (see page 239)	This is GetBaseLevel, a member of class TIBDatabaseInfo.
	GetCurrentMemory (see page 239)	This is GetCurrentMemory, a member of class TIBDatabaseInfo.
	GetDBFileName (see page 239)	This is GetDBFileName, a member of class TIBDatabaseInfo.
	GetDBImplementationClass (see page 240)	This is GetDBImplementationClass, a member of class TIBDatabaseInfo.
	GetDBImplementationNo (see page 240)	This is GetDBImplementationNo, a member of class TIBDatabaseInfo.
	GetDBSiteName (see page 240)	This is GetDBSiteName, a member of class TIBDatabaseInfo.
	GetDBSQLDialect (see page 240)	This is GetDBSQLDialect, a member of class TIBDatabaseInfo.
	GetDeleteCount (see page 241)	This is GetDeleteCount, a member of class TIBDatabaseInfo.
	GetExpungeCount (see page 241)	This is GetExpungeCount, a member of class TIBDatabaseInfo.
	GetFetches (see page 241)	This is GetFetches, a member of class TIBDatabaseInfo.
	GetForcedWrites (see page 241)	This is GetForcedWrites, a member of class TIBDatabaseInfo.
	GetInsertCount (see page 241)	This is GetInsertCount, a member of class TIBDatabaseInfo.
	GetMarks (see page 242)	This is GetMarks, a member of class TIBDatabaseInfo.
	GetMaxMemory (see page 242)	This is GetMaxMemory, a member of class TIBDatabaseInfo.
	GetNoReserve (see page 242)	This is GetNoReserve, a member of class TIBDatabaseInfo.
	GetNumBuffers (see page 243)	This is GetNumBuffers, a member of class TIBDatabaseInfo.
	GetODSMajorVersion (see page 243)	This is GetODSMajorVersion, a member of class TIBDatabaseInfo.
	GetODSMinorVersion (see page 243)	This is GetODSMinorVersion, a member of class TIBDatabaseInfo.
	GetOperationCounts (see page 243)	This is GetOperationCounts, a member of class TIBDatabaseInfo.
	GetPageSize (see page 244)	This is GetPageSize, a member of class TIBDatabaseInfo.
	GetPurgeCount (see page 244)	This is GetPurgeCount, a member of class TIBDatabaseInfo.
	GetReadIdxCount (see page 244)	This is GetReadIdxCount, a member of class TIBDatabaseInfo.
	GetReadOnly (see page 244)	This is GetReadOnly, a member of class TIBDatabaseInfo.
	GetReads (see page 244)	This is GetReads, a member of class TIBDatabaseInfo.
	GetReadSeqCount (see page 245)	This is GetReadSeqCount, a member of class TIBDatabaseInfo.
	GetStringDatabaseInfo (see page 245)	This is GetStringDatabaseInfo, a member of class TIBDatabaseInfo.
	GetSweepInterval (see page 245)	This is GetSweepInterval, a member of class TIBDatabaseInfo.
	GetUpdateCount (see page 245)	This is GetUpdateCount, a member of class TIBDatabaseInfo.
	GetUserNames (see page 246)	This is GetUserNames, a member of class TIBDatabaseInfo.
	GetVersion (see page 246)	This is GetVersion, a member of class TIBDatabaseInfo.
	GetWrites (see page 246)	This is GetWrites, a member of class TIBDatabaseInfo.

Public Methods

	Name	Description
	Call (see page 238)	Returns an error message based on the error code.
	GetLongDatabaseInfo (see page 242)	Returns a specified long integer that describes some aspect of the associated database.

6.1.1.5.1 TIBDatabaseInfo.Call Method

Returns an error message based on the error code.

Pascal

```
function Call(ErrCode: ISC_STATUS; RaiseError: Boolean): ISC_STATUS;
```

C++

```
__fastcall ISC_STATUS Call(ISC_STATUS ErrCode, Boolean RaiseError);
```

Description

Call is an internal method used to make calls to the InterBase API, and gives you the option of raising an exception or returning

an error based on the value of RaiseError.

6.1.1.5.2 TIBDatabaseInfo.GetAllocation Method

Pascal

```
function GetAllocation: Long;
```

C++

```
__fastcall Long GetAllocation();
```

Description

This is GetAllocation, a member of class TIBDatabaseInfo.

6.1.1.5.3 TIBDatabaseInfo.GetBackoutCount Method

Pascal

```
function GetBackoutCount: TStringList;
```

C++

```
__fastcall TStringList GetBackoutCount();
```

Description

This is GetBackoutCount, a member of class TIBDatabaseInfo.

6.1.1.5.4 TIBDatabaseInfo.GetBaseLevel Method

Pascal

```
function GetBaseLevel: Long;
```

C++

```
__fastcall Long GetBaseLevel();
```

Description

This is GetBaseLevel, a member of class TIBDatabaseInfo.

6.1.1.5.5 TIBDatabaseInfo.GetCurrentMemory Method

Pascal

```
function GetCurrentMemory: Long;
```

C++

```
__fastcall Long GetCurrentMemory();
```

Description

This is GetCurrentMemory, a member of class TIBDatabaseInfo.

6.1.1.5.6 TIBDatabaseInfo.GetDBFileName Method

Pascal

```
function GetDBFileName: String;
```

C++

```
__fastcall UnicodeString GetDBFileName();
```

Description

This is GetDBFileName, a member of class TIBDatabaseInfo.

6.1.1.5.7 TIBDatabaseInfo.GetDBImplementationClass Method

Pascal

```
function GetDBImplementationClass: Long;
```

C++

```
__fastcall Long GetDBImplementationClass();
```

Description

This is GetDBImplementationClass, a member of class TIBDatabaseInfo.

6.1.1.5.8 TIBDatabaseInfo.GetDBImplementationNo Method

Pascal

```
function GetDBImplementationNo: Long;
```

C++

```
__fastcall Long GetDBImplementationNo();
```

Description

This is GetDBImplementationNo, a member of class TIBDatabaseInfo.

6.1.1.5.9 TIBDatabaseInfo.GetDBSiteName Method

Pascal

```
function GetDBSiteName: String;
```

C++

```
__fastcall UnicodeString GetDBSiteName();
```

Description

This is GetDBSiteName, a member of class TIBDatabaseInfo.

6.1.1.5.10 TIBDatabaseInfo.GetDBSQLDialect Method

Pascal

```
function GetDBSQLDialect: Long;
```

C++

```
__fastcall Long GetDBSQLDialect();
```

Description

This is GetDBSQLDialect, a member of class TIBDatabaseInfo.

6.1.1.5.11 TIBDatabaseInfo.GetDeleteCount Method

Pascal

```
function GetDeleteCount: TStringList;
```

C++

```
__fastcall TStringList GetDeleteCount();
```

Description

This is GetDeleteCount, a member of class TIBDatabaseInfo.

6.1.1.5.12 TIBDatabaseInfo.GetExpungeCount Method

Pascal

```
function GetExpungeCount: TStringList;
```

C++

```
__fastcall TStringList GetExpungeCount();
```

Description

This is GetExpungeCount, a member of class TIBDatabaseInfo.

6.1.1.5.13 TIBDatabaseInfo.GetFetches Method

Pascal

```
function GetFetches: Long;
```

C++

```
__fastcall Long GetFetches();
```

Description

This is GetFetches, a member of class TIBDatabaseInfo.

6.1.1.5.14 TIBDatabaseInfo.GetForcedWrites Method

Pascal

```
function GetForcedWrites: Long;
```

C++

```
__fastcall Long GetForcedWrites();
```

Description

This is GetForcedWrites, a member of class TIBDatabaseInfo.

6.1.1.5.15 TIBDatabaseInfo.GetInsertCount Method

Pascal

```
function GetInsertCount: TStringList;
```

C++

```
__fastcall TStringList GetInsertCount();
```

Description

This is GetInsertCount, a member of class TIBDatabaseInfo.

6.1.1.5.16 TIBDatabaseInfo.GetLongDatabaseInfo Method

Returns a specified long integer that describes some aspect of the associated database.

Pascal

```
function GetLongDatabaseInfo(DatabaseInfoCommand: Integer): Long;
```

C++

```
__fastcall Long GetLongDatabaseInfo(int DatabaseInfoCommand);
```

Description

TIBDatabaseInfo (see page 232) uses GetLongDatabaseInfo internally to fetch information that can be expressed as a long integer. For example, this method provides the underlying implementation for fetching the version number or page size. It is exposed as a public member so that as the underlying _DatabaseInfoCommand interface expands its capabilities, TIBDatabaseInfo (see page 232) can still be used to fetch information for which there is no specified property.

DatabaseInfoCommand is an index that identifies the information to fetch. It is one of the constants declared in the IBHeader unit.

Long returns desired information.

6.1.1.5.17 TIBDatabaseInfo.GetMarks Method

Pascal

```
function GetMarks: Long;
```

C++

```
__fastcall Long GetMarks();
```

Description

This is GetMarks, a member of class TIBDatabaseInfo.

6.1.1.5.18 TIBDatabaseInfo.GetMaxMemory Method

Pascal

```
function GetMaxMemory: Long;
```

C++

```
__fastcall Long GetMaxMemory();
```

Description

This is GetMaxMemory, a member of class TIBDatabaseInfo.

6.1.1.5.19 TIBDatabaseInfo.GetNoReserve Method

Pascal

```
function GetNoReserve: Long;
```

C++

```
__fastcall Long GetNoReserve();
```

Description

This is GetNoReserve, a member of class TIBDatabaseInfo.

6.1.1.5.20 TIBDatabaseInfo.GetNumBuffers Method

Pascal

```
function GetNumBuffers: Long;
```

C++

```
__fastcall Long GetNumBuffers();
```

Description

This is GetNumBuffers, a member of class TIBDatabaseInfo.

6.1.1.5.21 TIBDatabaseInfo.GetODSMajorVersion Method

Pascal

```
function GetODSMajorVersion: Long;
```

C++

```
__fastcall Long GetODSMajorVersion();
```

Description

This is GetODSMajorVersion, a member of class TIBDatabaseInfo.

6.1.1.5.22 TIBDatabaseInfo.GetODSMinorVersion Method

Pascal

```
function GetODSMinorVersion: Long;
```

C++

```
__fastcall Long GetODSMinorVersion();
```

Description

This is GetODSMinorVersion, a member of class TIBDatabaseInfo.

6.1.1.5.23 TIBDatabaseInfo.GetOperationCounts Method

Pascal

```
function GetOperationCounts(DBInfoCommand: Integer; FOperation: TStringList): TStringList;
```

C++

```
__fastcall TStringList GetOperationCounts(int DBInfoCommand, TStringList FOperation);
```

Description

This is GetOperationCounts, a member of class TIBDatabaseInfo.

6.1.1.5.24 TIBDatabaseInfo.GetPageSize Method

Pascal

```
function GetPageSize: Long;
```

C++

```
__fastcall Long GetPageSize();
```

Description

This is GetPageSize, a member of class TIBDatabaseInfo.

6.1.1.5.25 TIBDatabaseInfo.GetPurgeCount Method

Pascal

```
function GetPurgeCount: TStringList;
```

C++

```
__fastcall TStringList GetPurgeCount();
```

Description

This is GetPurgeCount, a member of class TIBDatabaseInfo.

6.1.1.5.26 TIBDatabaseInfo.GetReadIdxCount Method

Pascal

```
function GetReadIdxCount: TStringList;
```

C++

```
__fastcall TStringList GetReadIdxCount();
```

Description

This is GetReadIdxCount, a member of class TIBDatabaseInfo.

6.1.1.5.27 TIBDatabaseInfo.GetReadOnly Method

Pascal

```
function GetReadOnly: Long;
```

C++

```
__fastcall Long GetReadOnly();
```

Description

This is GetReadOnly, a member of class TIBDatabaseInfo.

6.1.1.5.28 TIBDatabaseInfo.GetReads Method

Pascal

```
function GetReads: Long;
```

C++

```
__fastcall Long GetReads();
```

Description

This is GetReads, a member of class TIBDatabaseInfo.

6.1.1.5.29 TIBDatabaseInfo.GetReadSeqCount Method

Pascal

```
function GetReadSeqCount: TStringList;
```

C++

```
__fastcall TStringList GetReadSeqCount();
```

Description

This is GetReadSeqCount, a member of class TIBDatabaseInfo.

6.1.1.5.30 TIBDatabaseInfo.GetStringDatabaseInfo Method

Pascal

```
function GetStringDatabaseInfo(DatabaseInfoCommand: Integer): String;
```

C++

```
__fastcall UnicodeString GetStringDatabaseInfo(int DatabaseInfoCommand);
```

Description

This is GetStringDatabaseInfo, a member of class TIBDatabaseInfo.

6.1.1.5.31 TIBDatabaseInfo.GetSweepInterval Method

Pascal

```
function GetSweepInterval: Long;
```

C++

```
__fastcall Long GetSweepInterval();
```

Description

This is GetSweepInterval, a member of class TIBDatabaseInfo.

6.1.1.5.32 TIBDatabaseInfo.GetUpdateCount Method

Pascal

```
function GetUpdateCount: TStringList;
```

C++

```
__fastcall TStringList GetUpdateCount();
```

Description

This is GetUpdateCount, a member of class TIBDatabaseInfo.

6.1.1.5.33 TIBDatabaseInfo.GetUserNames Method

Pascal

```
function GetUserNames: TStringList;
```

C++

```
__fastcall TStringList GetUserNames();
```

Description

This is GetUserNames, a member of class TIBDatabaseInfo.

6.1.1.5.34 TIBDatabaseInfo.GetVersion Method

Pascal

```
function GetVersion: String;
```

C++

```
__fastcall UnicodeString GetVersion();
```

Description

This is GetVersion, a member of class TIBDatabaseInfo.

6.1.1.5.35 TIBDatabaseInfo.GetWrites Method

Pascal

```
function GetWrites: Long;
```

C++

```
__fastcall Long GetWrites();
```

Description

This is GetWrites, a member of class TIBDatabaseInfo.

6.1.1.6 TIBDatabaseInfo Properties

The properties of the TIBDatabaseInfo class are listed here.

Public Properties

	Name	Description
	Allocation (see page 247)	Returns the number of database pages allocated.
	BackoutCount (see page 247)	Returns the number of removals of a version of a record.
	BaseLevel (see page 248)	Returns the database version number.
	CurrentMemory (see page 248)	Returns the amount of server memory currently in use.
	DBFileName (see page 248)	Returns the database filename.
	DBImplementationClass (see page 249)	Returns the database implementation class number.
	DBImplementationNo (see page 249)	Returns the database implementation number.
	DBSiteName (see page 249)	Returns the database site name.
	DBSQLDialect (see page 249)	Returns the SQL dialect.
	DeleteCount (see page 250)	Returns the number of database deletes since the database was last attached.
	ExpungeCount (see page 250)	Returns the number of removals of a record and all of its ancestors.
	Fetches (see page 250)	Returns the number of reads from the memory buffer cache.

	ForcedWrites (see page 250)	Returns the mode in which database writes are performed.
	InsertCount (see page 251)	Returns number of inserts into the database since the database was last attached.
	Marks (see page 251)	Returns the number of writes to the memory buffer cache.
	MaxMemory (see page 251)	Returns the maximum amount of memory used at one time since the first process attached to the database.
	NoReserve (see page 251)	Returns whether or not space is reserved on each database page for holding backup versions of modified records.
	NumBuffers (see page 252)	Returns the number of memory buffers currently allocated.
	ODSMajorVersion (see page 252)	Returns the on disk structure (ODS) major version number.
	ODSMinorVersion (see page 252)	Returns the on disk structure (ODS) minor version number.
	PageSize (see page 252)	Returns the number of bytes per page of the attached database.
	PurgeCount (see page 253)	Returns the number of removals of fully mature records from the database.
	ReadIdxCount (see page 253)	Returns the number of reads done via an index since the database was last attached.
	ReadOnly (see page 253)	Indicates whether or not the database is read only.
	Reads (see page 253)	Returns the number of page reads from the database.
	ReadSeqCount (see page 254)	Returns the number of sequential database reads done on each table since the database was last attached.
	SweepInterval (see page 254)	Returns the number of transactions that are committed between 'sweeps.'
	UpdateCount (see page 254)	Returns the number of database updates since the database was last attached.
	UserNames (see page 255)	Returns the names of all users currently attached to the database.
	Version (see page 255)	Returns the version of the database implementation.
	Writes (see page 255)	Returns the number of page writes to the database.

Published Properties

	Name	Description
	Database (see page 248)	Sets or returns the database.

6.1.1.6.1 TIBDatabaseInfo.Allocation Property

Returns the number of database pages allocated.

Pascal

```
property Allocation: Long;
```

C++

```
__property Long Allocation;
```

Description

Use Allocation to return the number of database pages allocated.

6.1.1.6.2 TIBDatabaseInfo.BackoutCount Property

Returns the number of removals of a version of a record.

Pascal

```
property BackoutCount: TStringList;
```

C++

```
__property TStringList BackoutCount;
```

Description

Use BackoutCount to determine the number of times a version of a database record has been removed.

6.1.1.6.3 TIBDatabaseInfo.BaseLevel Property

Returns the database version number.

Pascal

```
property BaseLevel: Long;
```

C++

```
__property Long BaseLevel;
```

Description

Use BaseLevel to return the database version number, which consists of 1 byte containing the number 1, and 1 byte containing the version number.

6.1.1.6.4 TIBDatabaseInfo.CurrentMemory Property

Returns the amount of server memory currently in use.

Pascal

```
property CurrentMemory: Long;
```

C++

```
__property Long CurrentMemory;
```

Description

Use CurrentMemory to return the amount of server memory (in bytes) currently in use.

6.1.1.6.5 TIBDatabaseInfo.Database Property

Sets or returns the database.

Pascal

```
property Database: TIBDatabase;
```

C++

```
__property TIBDatabase Database;
```

Description

Use Database to set or return the database on which information is being returned.

6.1.1.6.6 TIBDatabaseInfo.DBFileName Property

Returns the database filename.

Pascal

```
property DBFileName: String;
```

C++

```
__property UnicodeString DBFileName;
```

Description

Use DBFileName to return the database filename.

6.1.1.6.7 TIBDatabaseInfo.DBImplementationClass Property

Returns the database implementation class number.

Pascal

```
property DBImplementationClass: Long;
```

C++

```
__property Long DBImplementationClass;
```

Description

Use DBImplementationClass to return the database implementation class number, either 1 or 12.

6.1.1.6.8 TIBDatabaseInfo.DBImplementationNo Property

Returns the database implementation number.

Pascal

```
property DBImplementationNo: Long;
```

C++

```
__property Long DBImplementationNo;
```

Description

Use DBImplementationNo to return the database implementation number.

6.1.1.6.9 TIBDatabaseInfo.DBSiteName Property

Returns the database site name.

Pascal

```
property DBSiteName: String;
```

C++

```
__property UnicodeString DBSiteName;
```

Description

Use DBSiteName to return the database site name.

6.1.1.6.10 TIBDatabaseInfo.DBSQLDialect Property

Returns the SQL dialect.

Pascal

```
property DBSQLDialect: Long;
```

C++

```
__property Long DBSQLDialect;
```

Description

Use DBSQLDialect to return the SQL dialect.

6.1.1.6.11 TIBDatabaseInfo.DeleteCount Property

Returns the number of database deletes since the database was last attached.

Pascal

```
property DeleteCount: TStringList;
```

C++

```
__property TStringList DeleteCount;
```

Description

Use DeleteCount to return the number of database deletes since the database was last attached.

6.1.1.6.12 TIBDatabaseInfo.ExpungeCount Property

Returns the number of removals of a record and all of its ancestors.

Pascal

```
property ExpungeCount: TStringList;
```

C++

```
__property TStringList ExpungeCount;
```

Description

Use ExpungeCount to return the number of removals of a record and all of its ancestors for records whose deletions have been committed.

6.1.1.6.13 TIBDatabaseInfo.Fetches Property

Returns the number of reads from the memory buffer cache.

Pascal

```
property Fetches: Long;
```

C++

```
__property Long Fetches;
```

Description

Use Fetches to return the number of reads from the memory buffer cache.

6.1.1.6.14 TIBDatabaseInfo.ForcedWrites Property

Returns the mode in which database writes are performed.

Pascal

```
property ForcedWrites: Long;
```

C++

```
__property Long ForcedWrites;
```

Description

Use ForcedWrites to return the number specifying the mode in which database writes are performed. ForcedWrites returns 0 for asynchronous mode, or returns 1 for synchronous mode

6.1.1.6.15 TIBDatabaseInfo.InsertCount Property

Returns number of inserts into the database since the database was last attached.

Pascal

```
property InsertCount: TStringList;
```

C++

```
__property TStringList InsertCount;
```

Description

Use InsertCount to return the number of inserts into the database since the database was last attached.

6.1.1.6.16 TIBDatabaseInfo.Marks Property

Returns the number of writes to the memory buffer cache.

Pascal

```
property Marks: Long;
```

C++

```
__property Long Marks;
```

Description

Use Marks to return the number of writes to the memory buffer cache.

6.1.1.6.17 TIBDatabaseInfo.MaxMemory Property

Returns the maximum amount of memory used at one time since the first process attached to the database.

Pascal

```
property MaxMemory: Long;
```

C++

```
__property Long MaxMemory;
```

Description

Use MaxMemory to return in bytes the maximum amount of memory used at one time since the first process attached to the database.

6.1.1.6.18 TIBDatabaseInfo.NoReserve Property

Returns whether or not space is reserved on each database page for holding backup versions of modified records.

Pascal

```
property NoReserve: Long;
```

C++

```
__property Long NoReserve;
```

Description

Use NoReserve to return whether or not space is reserved on each database page for holding backup versions of modified records. NoReserve will return 0 to indicate that space is reserved (the default) or 1 to indicate that no space is reserved.

6.1.1.6.19 TIBDatabaseInfo.NumBuffers Property

Returns the number of memory buffers currently allocated.

Pascal

```
property NumBuffers: Long;
```

C++

```
__property Long NumBuffers;
```

Description

Use NumBuffers to return the number of memory buffers currently allocated.

6.1.1.6.20 TIBDatabaseInfo.ODSMajorVersion Property

Returns the on disk structure (ODS) major version number.

Pascal

```
property ODSMajorVersion: Long;
```

C++

```
__property Long ODSMajorVersion;
```

Description

Use ODSMajorVersion to return the ODS major version number for the database. Databases with different major version numbers have different physical layouts.

A database engine can access only databases with a particular ODS major version number. Attaching to a database with a different ODS number results in an error.

6.1.1.6.21 TIBDatabaseInfo.ODSMinorVersion Property

Returns the on disk structure (ODS) minor version number.

Pascal

```
property ODSMinorVersion: Long;
```

C++

```
__property Long ODSMinorVersion;
```

Description

Use ODSMinorVersion to return the (ODS) minor version number. An increase in a minor version number indicates a non-structural change, one that still allows the database to be accessed by databases with the same major version number but possibly different minor version numbers.

6.1.1.6.22 TIBDatabaseInfo.PageSize Property

Returns the number of bytes per page of the attached database.

Pascal

```
property PageSize: Long;
```

C++

```
__property Long PageSize;
```

Description

Use PageSize to return the number of bytes per page of the attached database. Use with Allocation (see page 247) to determine the size of the database.

6.1.1.6.23 TIBDatabaseInfo.PurgeCount Property

Returns the number of removals of fully mature records from the database.

Pascal

```
property PurgeCount: TStringList;
```

C++

```
__property TStringList PurgeCount;
```

Description

Use PurgeCount to return the number of removals of fully mature records (that is, records committed, resulting in older versions no longer being needed) from the database.

6.1.1.6.24 TIBDatabaseInfo.ReadIdxCount Property

Returns the number of reads done via an index since the database was last attached.

Pascal

```
property ReadIdxCount: TStringList;
```

C++

```
__property TStringList ReadIdxCount;
```

Description

Use ReadIdxCount to return the number of reads done via an index since the database was last attached.

6.1.1.6.25 TIBDatabaseInfo.ReadOnly Property

Indicates whether or not the database is read only.

Pascal

```
property ReadOnly: Long;
```

C++

```
__property Long ReadOnly;
```

Description

Use ReadOnly to determine whether the database is read only or not. ReadOnly returns 1 if the database is read-write and 0 if it is read only.

Read-only databases are an InterBase 6 feature.

6.1.1.6.26 TIBDatabaseInfo.Reads Property

Returns the number of page reads from the database.

Pascal

```
property Reads: Long;
```

C++

```
__property Long Reads;
```

Description

Use Reads to return the number of page reads from the database since the current database was first attached, that is, an aggregate of all reads done by all attached processes, rather than the number of reads done for the calling program since it attached to the database.

6.1.1.6.27 TIBDatabaseInfo.ReadSeqCount Property

Returns the number of sequential database reads done on each table since the database was last attached.

Pascal

```
property ReadSeqCount: TStringList;
```

C++

```
__property TStringList ReadSeqCount;
```

Description

Use ReadSeqCount to return the number of sequential database reads (that is, the number of sequential table scans) done on each table since the database was last attached

6.1.1.6.28 TIBDatabaseInfo.SweepInterval Property

Returns the number of transactions that are committed between 'sweeps.'

Pascal

```
property SweepInterval: Long;
```

C++

```
__property Long SweepInterval;
```

Description

Use SweepInterval to return the number of transactions that are committed between 'sweeps' to remove database record versions that are no longer needed.

6.1.1.6.29 TIBDatabaseInfo.UpdateCount Property

Returns the number of database updates since the database was last attached.

Pascal

```
property UpdateCount: TStringList;
```

C++

```
__property TStringList UpdateCount;
```

Description

Use UpdateCount to return the number of database updates since the database was last attached.

6.1.1.6.30 TIBDatabaseInfo.UserNames Property

Returns the names of all users currently attached to the database.

Pascal

```
property UserNames: TStringList;
```

C++

```
__property TStringList UserNames;
```

Description

Use UserNames to return the names of all users currently attached to the database.

6.1.1.6.31 TIBDatabaseInfo.Version Property

Returns the version of the database implementation.

Pascal

```
property Version: String;
```

C++

```
__property UnicodeString Version;
```

Description

Use Version to return the version identification string of the database implementation.

6.1.1.6.32 TIBDatabaseInfo.Writes Property

Returns the number of page writes to the database.

Pascal

```
property Writes: Long;
```

C++

```
__property Long Writes;
```

Description

Use Writes to return the number of page writes to the current database since it was first attached by any process; that is, an aggregate of all write done by all attached processes, rather than the number of writes done for the calling program since it attached to the database.

7 IBEvents Namespace

This is namespace IBEvents.

Topics

Name	Description
Classes (see page 258)	The following table lists classes in this documentation.
Types (see page 266)	The following table lists types in this documentation.

Classes

	Name	Description
	TIBEvents (see page 258)	TIBEvents enables applications to respond to posted events.

Types

Name	Description
TErrorEvent (see page 266)	TErrorEvent is the type of event handlers that respond to exceptions that occur when monitoring InterBase events.
TEventAlert (see page 266)	TEventAlert is the type of event handlers that respond to InterBase events.

7.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBEvents (see page 258)	TIBEvents enables applications to respond to posted events.

7.1.1 TIBEvents Class

TIBEvents enables applications to respond to posted events.

Class Hierarchy

Pascal

```
TIBEvents = class(TComponent, IIBEventNotifier);
```

C++

```
class TIBEvents : public TComponent, public IIBEventNotifier;
```

File

IBEvents

Description

Use a TIBEvents component to allow your application to register interest in, and asynchronously handle, events posted by an InterBase server. The InterBase event allows applications to respond to actions and database changes made by other, concurrently running applications, without having to resort to polling the database on a regular basis, or communicating directly with the other applications.

In essence, the TIBEvents component allows an application to say 'I want to be informed when events X, Y and Z occur.' When any of the requested events does occur, the InterBase server notifies the application and OnEventAlert (see page 265) is called.

To use TIBEvents, set the Database (see page 263) property to the component that connects to the InterBase server. Then, use the Events (see page 264) property to indicate the events to which you want to respond. You can set the AutoRegister (see page 263) property to true if you want TIBEvents to automatically register interest in its events when the database component opens a connection, or set the Registered (see page 264) property to true after the connection is open to explicitly register your interest. Finally, write an OnEventAlert (see page 265) event handler to respond when the InterBase event handler informs TIBEvents that an event occurred.

7.1.1.1 TIBEvents Members

The following tables list the members exposed by TIBEvents.

Protected Methods

	Name	Description
	EventChange (see page 260)	This is EventChange, a member of class TIBEvents.
	GetNativeHandle (see page 261)	This is GetNativeHandle, a member of class TIBEvents.
	Notification (see page 261)	Forwards notification messages to all owned components.
	ThreadEnded (see page 262)	This is ThreadEnded, a member of class TIBEvents.
	ValidateDatabase (see page 263)	This is ValidateDatabase, a member of class TIBEvents.

Public Methods

	Name	Description
	Create (see page 259)	Creates an instance of TIBEvents (see page 258).
	Destroy (see page 260)	Destroys an instance of TIBEvents (see page 258).
	GetAutoRegister (see page 260)	Returns the value of the AutoRegister (see page 263) property.
	RegisterEvents (see page 261)	Registers interest in the events listed by the Events (see page 264) property.
	SetAutoRegister (see page 262)	Sets the AutoRegister (see page 263) property.
	UnRegisterEvents (see page 262)	Unregisters interest in the events listed by the Events (see page 264) property.

Public Properties

	Name	Description
	NativeHandle (see page 264)	Supplies the InterBase handle for the database on which events occur.

Published Events

	Name	Description
	OnError (see page 265)	Occurs when an exception is caught while monitoring events.
	OnEventAlert (see page 265)	Occurs when an InterBase event is received.

Published Properties

	Name	Description
	AutoRegister (see page 263)	Specifies the AutoRegister on which to perform event alerter tasks.
	Database (see page 263)	Specifies the database on which to perform event alerter tasks.
	Events (see page 264)	Specifies the events to which TIBEvents (see page 258) responds.
	Registered (see page 264)	Indicates whether the events in the Events (see page 264) property are registered.

7.1.1.2 TIBEvents.Create Constructor

Creates an instance of TIBEvents (see page 258).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBEvents(TComponent * AOwner);
```

Description

Call Create to create an instance of TIBEvents (see page 258). TIBEvents (see page 258) objects placed on a form or data module at design time are created automatically.

AOwner specifies the object (typically a form or data module) that is responsible for freeing the TIBEvents (see page 258) instance.

7.1.1.3 TIBEvents.Destroy Destructor

Destroys an instance of TIBEvents (see page 258).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBEvents();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TIBEvents (see page 258) object is not nil and only then calls Destroy. Destroy unregisters the events and then frees them.

7.1.1.4 TIBEvents Methods

The methods of the TIBEvents class are listed here.

Protected Methods

	Name	Description
	EventChange (see page 260)	This is EventChange, a member of class TIBEvents.
	GetNativeHandle (see page 261)	This is GetNativeHandle, a member of class TIBEvents.
	Notification (see page 261)	Forwards notification messages to all owned components.
	ThreadEnded (see page 262)	This is ThreadEnded, a member of class TIBEvents.
	ValidateDatabase (see page 263)	This is ValidateDatabase, a member of class TIBEvents.

Public Methods

	Name	Description
	GetAutoRegister (see page 260)	Returns the value of the AutoRegister (see page 263) property.
	RegisterEvents (see page 261)	Registers interest in the events listed by the Events (see page 264) property.
	SetAutoRegister (see page 262)	Sets the AutoRegister (see page 263) property.
	UnRegisterEvents (see page 262)	Unregisters interest in the events listed by the Events (see page 264) property.

7.1.1.4.1 TIBEvents.EventChange Method

Pascal

```
procedure EventChange(Sender: TObject); virtual;
```

C++

```
virtual __fastcall EventChange(TObject * Sender);
```

Description

This is EventChange, a member of class TIBEvents.

7.1.1.4.2 TIBEvents.GetAutoRegister Method

Returns the value of the AutoRegister (see page 263) property.

Pascal

```
function GetAutoRegister: Boolean;
```

C++

```
__fastcall Boolean GetAutoRegister();
```

Description

GetAutoRegister is the read implementation of the AutoRegister (see page 263) property.

7.1.1.4.3 TIBEvents.GetNativeHandle Method

Pascal

```
function GetNativeHandle: TISC_DB_HANDLE; virtual;
```

C++

```
virtual __fastcall TISC_DB_HANDLE GetNativeHandle();
```

Description

This is GetNativeHandle, a member of class TIBEvents.

7.1.1.4.4 TIBEvents.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

7.1.1.4.5 TIBEvents.RegisterEvents Method

Registers interest in the events listed by the Events (see page 264) property.

Pascal

```
procedure RegisterEvents; virtual;
```

C++

```
virtual __fastcall RegisterEvents();
```

Description

Call RegisterEvents to register interest in the events listed in the Events (see page 264) property with the InterBase Event Manager. Once this interest is registered, the InterBase Event Manager generates an OnEventAlert (see page 265) event when any of the registered events occurs.

Note: RegisterEvents is called automatically when you set the Registered (see page 264) property to true or when the AutoRegister (see page 263) property is true and the database connection is open.

7.1.1.4.6 TIBEvents.SetAutoRegister Method

Sets the AutoRegister (see page 263) property.

Pascal

```
procedure SetAutoRegister(const Value: Boolean);
```

C++

```
__fastcall SetAutoRegister(const Boolean Value);
```

Description

SetAutoRegister is the write implementation method for the AutoRegister (see page 263) property.

Value is the new value for the AutoRegister (see page 263) property.

When SetAutoRegister changes the AutoRegister (see page 263) property to true, it checks whether the database connection is already open. If so, it calls RegisterEvents (see page 261) to register the events.

7.1.1.4.7 TIBEvents.ThreadEnded Method

Pascal

```
procedure ThreadEnded(Sender: TObject); virtual;
```

C++

```
virtual __fastcall ThreadEnded(TObject * Sender);
```

Description

This is ThreadEnded, a member of class TIBEvents.

7.1.1.4.8 TIBEvents.UnRegisterEvents Method

Unregisters interest in the events listed by the Events (see page 264) property.

Pascal

```
procedure UnRegisterEvents; virtual;
```

C++

```
virtual __fastcall UnRegisterEvents();
```

Description

Call UnRegisterEvents to unregister interest in the events in the Events (see page 264) list. When the database closes a connection, or when the IBEvents (see page 257) component is destroyed, UnRegisterEvents is called automatically.

7.1.1.4.9 TIBEvents.ValidateDatabase Method

Pascal

```
procedure ValidateDatabase(DataBase: TIBDatabase); virtual;
```

C++

```
virtual __fastcall ValidateDatabase(TIBDatabase DataBase);
```

Description

This is ValidateDatabase, a member of class TIBEvents.

7.1.1.5 TIBEvents Properties

The properties of the TIBEvents class are listed here.

Public Properties

	Name	Description
	NativeHandle (see page 264)	Supplies the InterBase handle for the database on which events occur.

Published Properties

	Name	Description
	AutoRegister (see page 263)	Specifies the AutoRegister on which to perform event alerter tasks.
	Database (see page 263)	Specifies the database on which to perform event alerter tasks.
	Events (see page 264)	Specifies the events to which TIBEvents (see page 258) responds.
	Registered (see page 264)	Indicates whether the events in the Events (see page 264) property are registered.

7.1.1.5.1 TIBEvents.AutoRegister Property

Specifies the AutoRegister on which to perform event alerter tasks.

Pascal

```
property AutoRegister: Boolean;
```

C++

```
__property Boolean AutoRegister;
```

Description

When AutoRegister is true, the database automatically registers this component's events whenever it opens a connection to the InterBase server. When AutoRegister is false, you must explicitly register the events by setting the Registered (see page 264) property to true (or calling RegisterEvents (see page 261)) after the database connection is opened.

Note: The database always unregisters the events when it closes a connection. By setting AutoRegister to true, you do not need to explicitly re-register interest in the events when the database reopens a connection.

7.1.1.5.2 TIBEvents.Database Property

Specifies the database on which to perform event alerter tasks.

Pascal

```
property Database: TIBDatabase;
```

C++

```
__property TIBDatabase Database;
```

Description

Use Database to specify the component that manages a connection to the InterBase server. This connection must be open for TIBEvents (see page 258) to register an interest in its events with the InterBase Event manager on that server.

When you set the Database property, TIBEvents (see page 258) automatically calls the database component's AddEventNotifier method so that it can register this component's events when it opens a connection (if AutoRegister (see page 263) is true), and so that it can unregister this component's events before closing the connection.

7.1.1.5.3 TIBEvents.Events Property

Specifies the events to which TIBEvents (see page 258) responds.

Pascal

```
property Events: TStrings;
```

C++

```
__property TStrings Events;
```

Description

Use the Events property to list events to which the IBEvents (see page 257) component responds.

To add an event use code similar to the following:

```
Delphi: IBEvents.Events.Add('EVENT_NAME')
    C++: IBEvents->Events->Add("EVENT_NAME")
```

Note: Event names are case-sensitive.

7.1.1.5.4 TIBEvents.NativeHandle Property

Supplies the InterBase handle for the database on which events occur.

Pascal

```
property NativeHandle: TISC_DB_HANDLE;
```

C++

```
__property TISC_DB_HANDLE NativeHandle;
```

Description

Read NativeHandle to get the InterBase handle for the server on which the events occur. This property can be used when working with the InterBase API.

7.1.1.5.5 TIBEvents.Registered Property

Indicates whether the events in the Events (see page 264) property are registered.

Pascal

```
property Registered: Boolean;
```

C++

```
__property Boolean Registered;
```

Description

Set Registered to true to call RegisterEvents (see page 261), which registers the events listed by the Events (see page 264) property. Read Registered to determine whether TIBEvents (see page 258) is currently registered (that is, if the InterBase Event manager is notifying this component when any of its events occur.)

7.1.1.6 TIBEvents Events

The events of the TIBEvents class are listed here.

Published Events

	Name	Description
	OnError (see page 265)	Occurs when an exception is caught while monitoring events.
	OnEventAlert (see page 265)	Occurs when an InterBase event is received.

7.1.1.6.1 TIBEvents.OnError Event

Occurs when an exception is caught while monitoring events.

Pascal

```
property OnError: TErrorEvent;
```

C++

```
__property TErrorEvent OnError;
```

Description

Write an OnError event handler to respond when TIBEvents (see page 258) encounters an exception in the process of monitoring events.

7.1.1.6.2 TIBEvents.OnEventAlert Event

Occurs when an InterBase event is received.

Pascal

```
property OnEventAlert: TEventAlert;
```

C++

```
__property TEventAlert OnEventAlert;
```

Description

Write an OnEventAlert event handler to take specific actions when an InterBase event is received.

In the event handler, set CancelAlerts to true to cancel interest in any further events. To start receiving events again, call the GetAutoRegister (see page 260) method. You cannot call RegisterEvents (see page 261), UnRegisterEvents (see page 262), GetAutoRegister (see page 260) or SetAutoRegister (see page 262) from within an OnEventAlert event handler.

OnEventAlert runs as a separate thread to allow for true asynchronous event processing, however, the IBEvents (see page 257) component provides synchronization code to ensure that only one OnEventAlert event handler executes at any one time.

7.2 Types

The following table lists types in this documentation.

Types

Name	Description
TErrorEvent (see page 266)	TErrorEvent is the type of event handlers that respond to exceptions that occur when monitoring InterBase events.
TEventAlert (see page 266)	TEventAlert is the type of event handlers that respond to InterBase events.

7.2.1 IBEvents.TErrorEvent Type

TErrorEvent is the type of event handlers that respond to exceptions that occur when monitoring InterBase events.

Pascal

```
TErrorEvent = procedure ( Sender: TObject; ErrorCode:integer) of object;
```

C++

```
( Sender: TObject; ErrorCode:integer) ( TErrorEvent)();
```

File

IBEvents

Description

Sender is the TIBEvents (see page 258) component that encounters the exception.

If the exception is an EIBError (see page 3) object, ErrorCode is the InterBase error code it represents. Otherwise, ErrorCode is 0.

See Also

Procedural types

Creating events

7.2.2 IBEvents.TEventAlert Type

TEventAlert is the type of event handlers that respond to InterBase events.

Pascal

```
TEventAlert = procedure ( Sender: TObject; EventName: string; EventCount: longint; var
CancelAlerts: Boolean) of object;
```

C++

```
( Sender: TObject; EventName: string; EventCount: longint; var CancelAlerts: Boolean) (
TEventAlert)();
```

File

IBEvents

Description

Sender is the TIBEvents (see page 258) object that is monitoring InterBase events.

EventName is the name of the most recently received event.

EventCount is the number of EventName events received since the last time the event handler was called.

CancelAlerts returns true if the TIBEvents (see page 258) object should stop monitoring events, false if the event monitoring should continue.

8 IBExtract Namespace

This is namespace IBExtract.

Topics

Name	Description
Classes (see page 271)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 283)	The following table lists structs, records, enums in this documentation.
Types (see page 287)	The following table lists types in this documentation.
Constants (see page 289)	The following table lists constants in this documentation.

Classes

	Name	Description
	TIBExtract (see page 271)	TIBExtract fetches metadata from an InterBase server.

Constants

Name	Description
ColumnTypes (see page 290)	This is constant IBExtract.ColumnTypes.
FILE_cache (see page 290)	This is constant IBExtract.FILE_cache.
FILE_conditional (see page 290)	This is constant IBExtract.FILE_conditional.
FILE_inactive (see page 291)	This is constant IBExtract.FILE_inactive.
FILE_manual (see page 291)	This is constant IBExtract.FILE_manual.
FILE_shadow (see page 291)	This is constant IBExtract.FILE_shadow.
IntegralSubtypes (see page 292)	This is constant IBExtract.IntegralSubtypes.
LOG_default (see page 292)	This is constant IBExtract.LOG_default.
LOG_overflow (see page 292)	This is constant IBExtract.LOG_overflow.
LOG_raw (see page 293)	This is constant IBExtract.LOG_raw.
LOG_serial (see page 293)	This is constant IBExtract.LOG_serial.
MAX_INTSUBTYPES (see page 293)	This is constant IBExtract.MAX_INTSUBTYPES.
MAXSUBTYPES (see page 294)	This is constant IBExtract.MAXSUBTYPES.
obj_computed (see page 294)	This is constant IBExtract.obj_computed.
obj_count (see page 294)	This is constant IBExtract.obj_count.
obj_exception (see page 295)	This is constant IBExtract.obj_exception.
obj_expression_index (see page 295)	This is constant IBExtract.obj_expression_index.
obj_field (see page 295)	This is constant IBExtract.obj_field.
obj_index (see page 296)	This is constant IBExtract.obj_index.
obj_procedure (see page 296)	This is constant IBExtract.obj_procedure.

obj_relation (see page 296)	This is constant IBExtract.obj_relation.
obj_sql_role (see page 297)	This is constant IBExtract.obj_sql_role.
obj_trigger (see page 297)	This is constant IBExtract.obj_trigger.
obj_user (see page 297)	This is constant IBExtract.obj_user.
obj_user_group (see page 298)	This is constant IBExtract.obj_user_group.
obj_validation (see page 298)	This is constant IBExtract.obj_validation.
obj_view (see page 298)	This is constant IBExtract.obj_view.
ODS_VERSION10 (see page 299)	This is constant IBExtract.ODS_VERSION10.
ODS_VERSION6 (see page 299)	This is constant IBExtract.ODS_VERSION6.
ODS_VERSION7 (see page 299)	This is constant IBExtract.ODS_VERSION7.
ODS_VERSION8 (see page 300)	This is constant IBExtract.ODS_VERSION8.
ODS_VERSION9 (see page 300)	This is constant IBExtract.ODS_VERSION9.
priv_DELETE (see page 300)	This is constant IBExtract.priv_DELETE.
priv_EXECUTE (see page 301)	This is constant IBExtract.priv_EXECUTE.
priv_INSERT (see page 301)	This is constant IBExtract.priv_INSERT.
priv_REFERENCES (see page 301)	This is constant IBExtract.priv_REFERENCES.
priv_SELECT (see page 302)	This is constant IBExtract.priv_SELECT.
priv_UNKNOWN (see page 302)	This is constant IBExtract.priv_UNKNOWN.
priv_UPDATE (see page 302)	This is constant IBExtract.priv_UPDATE.
PrivTypes (see page 303)	This is constant IBExtract.PrivTypes.
SubTypes (see page 303)	This is constant IBExtract.SubTypes.
TriggerTypes (see page 303)	This is constant IBExtract.TriggerTypes.

Structs, Records, Enums

	Name	Description
	TExtractObjectTypes (see page 283)	TExtractObjectTypes indicates the type of metadata to fetch from an InterBase server.
	TExtractType (see page 284)	TExtractType is an enumeration indicating the scope of metadata to extract from an InterBase server.
	TPrivTypes (see page 285)	This is record IBExtract.TPrivTypes.
	TSQLType (see page 286)	This is record IBExtract.TSQLType.

Types

Name	Description
TExtractTypes (see page 287)	TExtractType (see page 284) and TExtractTypes indicate the scope of metadata to extract from an InterBase server.
TSQLTypes (see page 288)	This is type IBExtract.TSQLTypes.

8.1 Classes

The following table lists classes in this documentation.

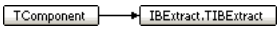
Classes

	Name	Description
	TIBExtract (see page 271)	TIBExtract fetches metadata from an InterBase server.

8.1.1 TIBExtract Class

TIBExtract fetches metadata from an InterBase server.

Class Hierarchy



Pascal

```
TIBExtract = class(TComponent);
```

C++

```
class TIBExtract : public TComponent;
```

File

IBExtract

Description

Use a TIBExtract component to fetch metadata information such as a list of tables, views, roles, indexes, and so on. To use TIBExtract,

1. et the Database (see page 281) and Transaction (see page 282) properties to specify the database from which to fetch information and to provide a transaction to use when fetching that information.
2. all the ExtractObject (see page 274) method, indicating the type of metadata you want to fetch.
3. ead the extracted information from the Items (see page 281) property.

See Also

Getting Started with InterBase Express

8.1.1.1 TIBExtract Members

The following tables list the members exposed by TIBExtract.

Protected Methods

	Name	Description
	ExtractDDL (see page 274)	This is ExtractDDL, a member of class TIBExtract.
	ExtractListTable (see page 274)	This is ExtractListTable, a member of class TIBExtract.
	ExtractListView (see page 274)	This is ExtractListView, a member of class TIBExtract.
	ListAllTables (see page 276)	This is ListAllTables, a member of class TIBExtract.

	ListCheck (see page 277)	This is ListCheck, a member of class TIBExtract.
	ListCreateDb (see page 277)	This is ListCreateDb, a member of class TIBExtract.
	ListData (see page 277)	This is ListData, a member of class TIBExtract.
	ListDomains (see page 277)	This is ListDomains, a member of class TIBExtract.
	ListException (see page 278)	This is ListException, a member of class TIBExtract.
	ListFilters (see page 278)	This is ListFilters, a member of class TIBExtract.
	ListForeign (see page 278)	This is ListForeign, a member of class TIBExtract.
	ListFunctions (see page 278)	This is ListFunctions, a member of class TIBExtract.
	ListGenerators (see page 278)	This is ListGenerators, a member of class TIBExtract.
	ListGrants (see page 279)	This is ListGrants, a member of class TIBExtract.
	ListIndex (see page 279)	This is ListIndex, a member of class TIBExtract.
	ListProcs (see page 279)	This is ListProcs, a member of class TIBExtract.
	ListRoles (see page 279)	This is ListRoles, a member of class TIBExtract.
	ListTriggers (see page 280)	This is ListTriggers, a member of class TIBExtract.
	ListViews (see page 280)	This is ListViews, a member of class TIBExtract.
	PrintSet (see page 280)	This is PrintSet, a member of class TIBExtract.

Public Methods

	Name	Description
V	Create (see page 272)	Creates an instance of TIBExtract (see page 271).
V	Destroy (see page 273)	Destroys an instance of TIBExtract (see page 271).
	ExtractObject (see page 274)	Fetches specified metadata from the InterBase server and writes it to the Items (see page 281) property.
	GetArrayField (see page 276)	Returns the dimensions of an array field as a printable string.
	GetCharacterSets (see page 276)	Returns a specified character set and collation order as a printable string.
	GetFieldType (see page 276)	Returns the full description of a specified field type as a printable string.
V	Notification (see page 280)	Responds to notifications when components are created or freed.

Public Properties

	Name	Description
R	DatabaseInfo (see page 281)	Represents information about the associated database.
R	Items (see page 281)	Lists the information that the TIBExtract (see page 271) object has fetched from the database server.

Published Properties

	Name	Description
	Database (see page 281)	Specifies the database from which to fetch metadata.
	ShowSystem (see page 282)	Reserved for future use.
	Transaction (see page 282)	Specifies the transaction to use when fetching metadata from the server.

8.1.1.2 TIBExtract.Create Constructor

Creates an instance of TIBExtract (see page 271).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBExtract(TComponent * AOwner);
```

Description

Call Create to create an instance of TIBExtract (see page 271) at runtime. TIBExtract (see page 271) components added to forms or data modules at design time are created automatically.

AOwner specifies the component that is responsible for freeing the TIBExtract (see page 271) instance. If AOwner is a

TIBDatabase component, it is used to initialize the Database (see page 281) property.

8.1.1.3 TIBExtract.Destroy Destructor

Destroys an instance of TIBExtract (see page 271).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBExtract();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TIBExtract (see page 271) object is not already freed and only then calls Destroy.

There is no need to explicitly free a TIBExtract (see page 271) instance that is created with an Owner. The component's Owner handles the task of freeing it.

8.1.1.4 TIBExtract Methods

The methods of the TIBExtract class are listed here.

Protected Methods

	Name	Description
	ExtractDDL (see page 274)	This is ExtractDDL, a member of class TIBExtract.
	ExtractListTable (see page 274)	This is ExtractListTable, a member of class TIBExtract.
	ExtractListView (see page 274)	This is ExtractListView, a member of class TIBExtract.
	ListAllTables (see page 276)	This is ListAllTables, a member of class TIBExtract.
	ListCheck (see page 277)	This is ListCheck, a member of class TIBExtract.
	ListCreateDb (see page 277)	This is ListCreateDb, a member of class TIBExtract.
	ListData (see page 277)	This is ListData, a member of class TIBExtract.
	ListDomains (see page 277)	This is ListDomains, a member of class TIBExtract.
	ListException (see page 278)	This is ListException, a member of class TIBExtract.
	ListFilters (see page 278)	This is ListFilters, a member of class TIBExtract.
	ListForeign (see page 278)	This is ListForeign, a member of class TIBExtract.
	ListFunctions (see page 278)	This is ListFunctions, a member of class TIBExtract.
	ListGenerators (see page 278)	This is ListGenerators, a member of class TIBExtract.
	ListGrants (see page 279)	This is ListGrants, a member of class TIBExtract.
	ListIndex (see page 279)	This is ListIndex, a member of class TIBExtract.
	ListProcs (see page 279)	This is ListProcs, a member of class TIBExtract.
	ListRoles (see page 279)	This is ListRoles, a member of class TIBExtract.
	ListTriggers (see page 280)	This is ListTriggers, a member of class TIBExtract.
	ListViews (see page 280)	This is ListViews, a member of class TIBExtract.
	PrintSet (see page 280)	This is PrintSet, a member of class TIBExtract.

Public Methods

	Name	Description
	ExtractObject (see page 274)	Fetches specified metadata from the InterBase server and writes it to the Items (see page 281) property.
	GetArrayField (see page 276)	Returns the dimensions of an array field as a printable string.
	GetCharacterSets (see page 276)	Returns a specified character set and collation order as a printable string.
	GetFieldType (see page 276)	Returns the full description of a specified field type as a printable string.

	Notification (see page 280)	Responds to notifications when components are created or freed.

8.1.1.4.1 TIBExtract.ExtractDDL Method

Pascal

```
function ExtractDDL(Flag: Boolean; TableName: String): Boolean;
```

C++

```
__fastcall Boolean ExtractDDL(Boolean Flag, UnicodeString TableName);
```

Description

This is ExtractDDL, a member of class TIBExtract.

8.1.1.4.2 TIBExtract.ExtractListTable Method

Pascal

```
function ExtractListTable(RelationName: String; NewName: String; DomainFlag: Boolean): Boolean;
```

C++

```
__fastcall Boolean ExtractListTable(UnicodeString RelationName, UnicodeString NewName, Boolean
DomainFlag);
```

Description

This is ExtractListTable, a member of class TIBExtract.

8.1.1.4.3 TIBExtract.ExtractListView Method

Pascal

```
procedure ExtractListView(ViewName: String);
```

C++

```
__fastcall ExtractListView(UnicodeString ViewName);
```

Description

This is ExtractListView, a member of class TIBExtract.

8.1.1.4.4 TIBExtract.ExtractObject Method

Fetches specified metadata from the InterBase server and writes it to the Items (see page 281) property.

Pascal

```
procedure ExtractObject(ObjectType: TExtractObjectTypes; ObjectName: String = '';
ExtractTypes: TExtractTypes = []);
```

C++

```
__fastcall ExtractObject(TExtractObjectTypes ObjectType, UnicodeString ObjectName = '',
TExtractTypes ExtractTypes = []);
```

Description

Call ExtractObject to fetch metadata from the database specified by the Database (see page 281) property and write it to the Items (see page 281) property.

ObjectType indicates the desired type of metadata. Its meaning is modified by the ObjectName and ExtractTypes parameters, as

indicated in the following table:

Value Meaning

eoDatabase Lists all objects defined on the database. This is the list that would be fetched by calling ExtractObject for each of the other object types in turn (plus the create database command that is supported), with no limits placed by the ObjectName and ExtractTypes parameters.

eoDomain Lists all domains. If ExtractTypes includes etTable, this list includes only domains for the table specified by ObjectName.

eoTable If ObjectName is an empty string, lists all the tables defined on the server. If ObjectName is non-empty, lists all the columns and types of the table named by ObjectName, plus any information specified by ExtractTypes that pertains to that table. This can include domains (etDomain), indexes (etIndex), foreign key constrains (etForeign), check constrains (etCheck), triggers (etTrigger), user permissions (etGrant), and data (etData).

eoView If ObjectName is an empty string, lists the text of all views. Otherwise, lists the text of the view named by the ObjectName parameter.

eoProcedure If ObjectName is an empty string, lists all the stored procedures. Otherwise, lists the text of the stored procedure named by the ObjectName parameter. To handle the situation where procedures reference each other, ExtractObject creates dummy procedures of the correct name and then alters these to their correct form. If ExtractTypes includes etAlterProc, only the corrected form is fetched.

eoFunction Lists all external functions. If ObjectName is non-empty, the list is filtered to include only those functions that match the specified name.

eoGenerator Lists all non-system generators. If ObjectName is non-empty, the list is filtered to include only those generators that match the specified name.

eoException Lists all exceptions that the database defines. If ObjectName is non-empty, the list is filtered to include only those exceptions that match the specified name.

eoBLOBFilter Lists all blob filters. If ObjectName is non-empty, the list is filtered to include only those filters whose function name matches the specified name.

eoRole Lists all roles. If ObjectName is non-empty, the list is filtered to include only those roles whose name matches the specified string.

eoTrigger If ExtractTypes includes etTable, lists all triggers defined for the table specified by ObjectName. Otherwise, lists all triggers or, if ObjectName is nonempty, all triggers that match the string specified by ObjectName.

eoForeign If ExtractTypes includes etTable, lists all foreign key constraints that pertain to the table specified by ObjectName. Otherwise, list all foreign key constraints, or, if ObjectName is nonempty, all foreign key constraints that have a name that matches ObjectName.

eoIndexes If ExtractTypes includes etTable, lists all indexes defined on the table specified by ObjectName. Otherwise, lists all indexes, or, if ObjectName is nonempty, all indexes that have a name that matches ObjectName.

eoChecks If ExtractTypes includes etTable, lists all non-foreign check constraints on the table specified by ObjectName. Otherwise, lists all non-foreign check constraints, or, if ObjectName is nonempty, all check constraints that have a name that matches ObjectName.

eoData Lists all the data in the object named by the ObjectName parameter.

ObjectName specifies the name of an object (such as a table or stored procedure) about which you want information. The type of this object and the way it limits the fetched information depends on the values of ObjectType and ExtractTypes.

ExtractTypes modifies the interpretation of ObjectType, to filter the type of information fetched.

8.1.1.4.5 TIBExtract.GetArrayField Method

Returns the dimensions of an array field as a printable string.

Pascal

```
function GetArrayField(FieldName: String): String;
```

C++

```
__fastcall UnicodeString GetArrayField(UnicodeString FieldName);
```

Description

GetArrayField returns the dimensions of the array field with the name specified by the FieldName parameter. The resulting string has the form

```
[lower:upper, lower:upper,..]
```

8.1.1.4.6 TIBExtract.GetCharacterSets Method

Returns a specified character set and collation order as a printable string.

Pascal

```
function GetCharacterSets(CharSetId: Short; Collation: Short; CollateOnly: Boolean): String;
```

C++

```
__fastcall UnicodeString GetCharacterSets(Short CharSetId, Short Collation, Boolean
CollateOnly);
```

Description

GetCharacterSets looks up a specified character set and collation order, and returns their names.

CharSetId is the identifier of the desired character set.

Collation is the identifier of the desired collation order.

CollateOnly indicates that the return string should include the name of the collation order but not the name of the character set.

8.1.1.4.7 TIBExtract.GetFieldType Method

Returns the full description of a specified field type as a printable string.

Pascal

```
function GetFieldType(FieldType: Integer; FieldSubType: Integer; FieldScale: Integer;
FieldSize: Integer; FieldPrec: Integer; FieldLen: Integer): String;
```

C++

```
__fastcall UnicodeString GetFieldType(int FieldType, int FieldSubType, int FieldScale, int
FieldSize, int FieldPrec, int FieldLen);
```

Description

GetFieldType returns the string description with the specified field type, subtype, scale, size, precision, and length.

8.1.1.4.8 TIBExtract.ListAllTables Method

Pascal

```
procedure ListAllTables(flag: Boolean);
```

C++

```
__fastcall ListAllTables(Boolean flag);
```

Description

This is ListAllTables, a member of class TIBExtract.

8.1.1.4.9 TIBExtract.ListCheck Method

Pascal

```
procedure ListCheck(ObjectName: String = ''; ExtractType: TExtractType = etCheck);
```

C++

```
__fastcall ListCheck(UnicodeString ObjectName = '', TExtractType ExtractType = etCheck);
```

Description

This is ListCheck, a member of class TIBExtract.

8.1.1.4.10 TIBExtract.ListCreateDb Method

Pascal

```
procedure ListCreateDb(TargetDb: String = '');
```

C++

```
__fastcall ListCreateDb(UnicodeString TargetDb = '');
```

Description

This is ListCreateDb, a member of class TIBExtract.

8.1.1.4.11 TIBExtract.ListData Method

Pascal

```
procedure ListData(ObjectName: String);
```

C++

```
__fastcall ListData(UnicodeString ObjectName);
```

Description

This is ListData, a member of class TIBExtract.

8.1.1.4.12 TIBExtract.ListDomains Method

Pascal

```
procedure ListDomains(ObjectName: String = ''; ExtractType: TExtractType = etDomain);
```

C++

```
__fastcall ListDomains(UnicodeString ObjectName = '', TExtractType ExtractType = etDomain);
```

Description

This is ListDomains, a member of class TIBExtract.

8.1.1.4.13 TIBExtract.ListException Method

Pascal

```
procedure ListException(ExceptionName: String = '');
```

C++

```
__fastcall ListException(UnicodeString ExceptionName = '');
```

Description

This is ListException, a member of class TIBExtract.

8.1.1.4.14 TIBExtract.ListFilters Method

Pascal

```
procedure ListFilters(FilterName: String = '');
```

C++

```
__fastcall ListFilters(UnicodeString FilterName = '');
```

Description

This is ListFilters, a member of class TIBExtract.

8.1.1.4.15 TIBExtract.ListForeign Method

Pascal

```
procedure ListForeign(ObjectName: String = ''; ExtractType: TExtractType = etForeign);
```

C++

```
__fastcall ListForeign(UnicodeString ObjectName = '', TExtractType ExtractType = etForeign);
```

Description

This is ListForeign, a member of class TIBExtract.

8.1.1.4.16 TIBExtract.ListFunctions Method

Pascal

```
procedure ListFunctions(FunctionName: String = '');
```

C++

```
__fastcall ListFunctions(UnicodeString FunctionName = '');
```

Description

This is ListFunctions, a member of class TIBExtract.

8.1.1.4.17 TIBExtract.ListGenerators Method

Pascal

```
procedure ListGenerators(GeneratorName: String = '');
```

C++

```
__fastcall ListGenerators(UnicodeString GeneratorName = '');
```

Description

This is ListGenerators, a member of class TIBExtract.

8.1.1.4.18 TIBExtract.ListGrants Method

Pascal

```
procedure ListGrants;
```

C++

```
__fastcall ListGrants();
```

Description

This is ListGrants, a member of class TIBExtract.

8.1.1.4.19 TIBExtract.ListIndex Method

Pascal

```
procedure ListIndex(ObjectName: String = ''; ExtractType: TExtractType = etIndex);
```

C++

```
__fastcall ListIndex(UnicodeString ObjectName = '', TExtractType ExtractType = etIndex);
```

Description

This is ListIndex, a member of class TIBExtract.

8.1.1.4.20 TIBExtract.ListProcs Method

Pascal

```
procedure ListProcs(ProcedureName: String = ''; AlterOnly: Boolean = false);
```

C++

```
__fastcall ListProcs(UnicodeString ProcedureName = '', Boolean AlterOnly = false);
```

Description

This is ListProcs, a member of class TIBExtract.

8.1.1.4.21 TIBExtract.ListRoles Method

Pascal

```
procedure ListRoles(ObjectName: String = '');
```

C++

```
__fastcall ListRoles(UnicodeString ObjectName = '');
```

Description

This is ListRoles, a member of class TIBExtract.

8.1.1.4.22 TIBExtract.ListTriggers Method

Pascal

```
procedure ListTriggers(ObjectName: String = ''; ExtractType: TExtractType = etTrigger);
```

C++

```
__fastcall ListTriggers(UnicodeString ObjectName = '', TExtractType ExtractType = etTrigger);
```

Description

This is ListTriggers, a member of class TIBExtract.

8.1.1.4.23 TIBExtract.ListViews Method

Pascal

```
procedure ListViews(ViewName: String = '');
```

C++

```
__fastcall ListViews(UnicodeString ViewName = '');
```

Description

This is ListViews, a member of class TIBExtract.

8.1.1.4.24 TIBExtract.Notification Method

Responds to notifications when components are created or freed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification is called automatically when objects are created or freed so that TIBExtract (see page 271) can make any necessary adjustments.

AComponent is the component that was just created or that is about to be destroyed.

Operation indicates whether AComponent was just created (opInsert) or is about to be freed (opRemove).

In TIBExtract (see page 271), Notification checks whether the components that implement the Database (see page 281) and Transaction (see page 282) properties are about to be freed, and if so, sets the appropriate property to nil (Delphi) or NULL (C++).

8.1.1.4.25 TIBExtract.PrintSet Method

Pascal

```
function PrintSet(var Used: Boolean): String;
```

C++

```
__fastcall UnicodeString PrintSet(Boolean Used);
```

Description

This is PrintSet, a member of class TIBExtract.

8.1.1.5 TIBExtract Properties

The properties of the TIBExtract class are listed here.

Public Properties

	Name	Description
	DatabaseInfo (see page 281)	Represents information about the associated database.
	Items (see page 281)	Lists the information that the TIBExtract (see page 271) object has fetched from the database server.

Published Properties

	Name	Description
	Database (see page 281)	Specifies the database from which to fetch metadata.
	ShowSystem (see page 282)	Reserved for future use.
	Transaction (see page 282)	Specifies the transaction to use when fetching metadata from the server.

8.1.1.5.1 TIBExtract.Database Property

Specifies the database from which to fetch metadata.

Pascal

```
property Database: TIBDatabase;
```

C++

```
__property TIBDatabase Database;
```

Description

Use Database to specify the database from which to fetch metadata.

8.1.1.5.2 TIBExtract.DatabaseInfo Property

Represents information about the associated database.

Pascal

```
property DatabaseInfo: TIBDatabaseInfo;
```

C++

```
__property TIBDatabaseInfo DatabaseInfo;
```

Description

Read DatabaseInfo to obtain information about the database that is specified by the Database (see page 281) property. DatabaseInfo provides a variety of information about the database itself, such as the version of the online disk structure (ODS) used by the attachment, the number of database cache buffers allocated, the number of database pages read from or written to, and write-ahead log information.

8.1.1.5.3 TIBExtract.Items Property

Lists the information that the TIBExtract (see page 271) object has fetched from the database server.

Pascal

```
property Items: TStrings;
```

C++

```
__property TStrings Items;
```

Description

Use the Items property to read the metadata information that was fetched from the database server. The ExtractObject (see page 274) method writes information to the Items property every time you call it.

Tip: TIBExtract (see page 271) does not clear the Items property when you make a new call to ExtractObject (see page 274). You may want to clear this list of strings between calls to ExtractObject (see page 274) so that the list does not get too long.

8.1.1.5.4 TIBExtract.ShowSystem Property

Reserved for future use.

Pascal

```
property ShowSystem: Boolean;
```

C++

```
__property Boolean ShowSystem;
```

Description

Currently, ShowSystem is not implemented.

8.1.1.5.5 TIBExtract.Transaction Property

Specifies the transaction to use when fetching metadata from the server.

Pascal

```
property Transaction: TIBTransaction;
```

C++

```
__property TIBTransaction Transaction;
```

Description

Set Transaction to specify a transaction to use when fetching metadata from the server. The ExtractObject (see page 274) method executes all calls to the server within the context of this transaction.

8.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TExtractObjectTypes (see page 283)	TExtractObjectTypes indicates the type of metadata to fetch from an InterBase server.
	TExtractType (see page 284)	TExtractType is an enumeration indicating the scope of metadata to extract from an InterBase server.

Records

	Name	Description
	TPrivTypes (see page 285)	This is record IBExtract.TPrivTypes.
	TSQLType (see page 286)	This is record IBExtract.TSQLType.

8.2.1 IBExtract.TExtractObjectTypes Enumeration

TExtractObjectTypes indicates the type of metadata to fetch from an InterBase server.

Pascal

```
TExtractObjectTypes = (
  eoDatabase,
  eoDomain,
  eoTable,
  eoView,
  eoProcedure,
  eoFunction,
  eoGenerator,
  eoException,
  eoBLOBFilter,
  eoRole,
  eoTrigger,
  eoForeign,
  eoIndexes,
  eoChecks,
  eoData
);
```

C++

```
enum TExtractObjectTypes {
  eoDatabase,
  eoDomain,
  eoTable,
  eoView,
  eoProcedure,
  eoFunction,
  eoGenerator,
  eoException,
  eoBLOBFilter,
  eoRole,
  eoTrigger,
  eoForeign,
  eoIndexes,
```

```
    eoChecks,
    eoData
  };
```

File

IBExtract

Description

Value Meaning

eoDatabase Fetch metadata about all objects defined on the database.

eoDomain List all domains, either for the entire database or only those that pertain to a specified table.

eoTable List all tables or information about a specified table.

eoView List all views in the database or the text of a specified view.

eoProcedure List all stored procedures or the text of a specified stored procedure.

eoFunction List all external functions, or list those external functions that match a specified name.

eoGenerator List all non-system generators, or only those generators that match a specified name.

eoException List all exceptions that the database defines, or only those exceptions that match a specified name.

eoBLOBFilter List all blob filters or only those filters whose function name matches a specified name.

eoRole List all roles or only those roles whose name matches a specified string.

eoTrigger List all triggers. The scope can be the entire database, those triggers that match a specified string, or those triggers defined for a specified table.

eoForeign List all foreign key constraints. The scope can be the entire database, those foreign key constraints with a specified name, or those that pertain to a specified table.

eoIndexes List all indexes. The scope can be the entire database, those indexes with a specified name, or those that are defined for a specified table.

eoChecks List all non-foreign check constraints. The scope can be the entire database, those check constraints with a specified name, or those that pertain to a specified table.

eoData List all the data a specified object.

8.2.2 IBExtract.TExtractType Enumeration

TExtractType is an enumeration indicating the scope of metadata to extract from an InterBase server.

Pascal

```
  TExtractType = (
    etDomain,
    etTable,
    etRole,
    etTrigger,
    etForeign,
    etIndex,
    etData,
    etGrant,
    etCheck,
    etAlterProc
```

```
);
```

C++

```
enum TExtractType {
  etDomain,
  etTable,
  etRole,
  etTrigger,
  etForeign,
  etIndex,
  etData,
  etGrant,
  etCheck,
  etAlterProc
};
```

File

IBExtract

Description

The following table lists the TExtractType enum items and meanings:

Value	Meaning
etDomain	When fetching metadata about a database table, include information about all domains that pertain to the table.
etTable	Fetch metadata of the target type that pertains only to a specified named table.
etRole	When fetching metadata about a database table, include information about all roles that have access to the table.
etTrigger	When fetching metadata about a database table, include information about all triggers that pertain to the table.
etForeign	When fetching metadata about a database table, include information about foreign key constraints.
etIndex	When fetching metadata about a database table, include information about indexes defined for the table.
etData	When fetching metadata about a database table, include the data values.
etGrant	When fetching metadata about a database table, include information about user permissions.
etCheck	When fetching metadata about a database table, include information about check constraints.
etAlterProc	When fetching metadata about stored procedures, the extraction may involve the creation of dummy procedures to handle the situation where procedures reference each other. These dummy procedures, which stand in for a referenced stored procedure, are later altered to their correct form. This flag indicates that only the corrected form should be fetched.

8.2.3 IBExtract.TPrivTypes Record

Pascal

```
TPrivTypes = record
  PrivFlag: Integer;
  PrivString: String;
end;
```

C++

```
struct TPrivTypes {
  int PrivFlag;
  UnicodeString PrivString;
};
```

File

IBExtract

Description

This is record IBExtract.TPrivTypes.

8.2.4 IBExtract.TSQLType Record

Pascal

```
TSQLType = record
  SqlType: Integer;
  TypeName: String;
end;
```

C++

```
struct TSQLType {
  int SqlType;
  UnicodeString TypeName;
};
```

File

IBExtract

Description

This is record IBExtract.TSQLType.

8.3 Types

The following table lists types in this documentation.

Types

Name	Description
TExtractTypes (see page 287)	TExtractType (see page 284) and TExtractTypes indicate the scope of metadata to extract from an InterBase server.
TSQLTypes (see page 288)	This is type IBExtract.TSQLTypes.

8.3.1 IBExtract.TExtractTypes Type

TExtractType (see page 284) and TExtractTypes indicate the scope of metadata to extract from an InterBase server.

Pascal

```
TExtractTypes = set of TExtractType;
```

C++

```
TExtractType TExtractTypes;
```

File

IBExtract

Description

TExtractTypes is a set of TExtractType (see page 284) values. The following table lists the possible values:

Value	Meaning
etDomain	When fetching metadata about a database table, include information about all domains that pertain to the table.
etTable	Fetch metadata of the target type that pertains only to a specified named table.
etRole	When fetching metadata about a database table, include information about all roles that have access to the table.
etTrigger	When fetching metadata about a database table, include information about all triggers that pertain to the table.
etForeign	When fetching metadata about a database table, include information about foreign key constraints.
etIndex	When fetching metadata about a database table, include information about indexes defined for the table.
etData	When fetching metadata about a database table, include the data values.
etGrant	When fetching metadata about a database table, include information about user permissions.
etCheck	When fetching metadata about a database table, include information about check constraints.
etAlterProc	When fetching metadata about stored procedures, the extraction may involve the creation of dummy procedures to handle the situation where procedures reference each other. These dummy procedures, which stand in for a referenced stored procedure, are later altered to their correct form. This flag indicates that only the corrected form should be fetched.

8.3.2 IBExtract.TSQLTypes Type

Pascal

```
TSQLTypes = Array[0..13] of TSQLType;
```

C++

```
Array[0..13] of TSQLType TSQLTypes;
```

File

IBExtract

Description

This is type IBExtract.TSQLTypes.

8.4 Constants

The following table lists constants in this documentation.

Constants

Name	Description
ColumnTypes (see page 290)	This is constant IBExtract.ColumnTypes.
FILE_cache (see page 290)	This is constant IBExtract.FILE_cache.
FILE_conditional (see page 290)	This is constant IBExtract.FILE_conditional.
FILE_inactive (see page 291)	This is constant IBExtract.FILE_inactive.
FILE_manual (see page 291)	This is constant IBExtract.FILE_manual.
FILE_shadow (see page 291)	This is constant IBExtract.FILE_shadow.
IntegralSubtypes (see page 292)	This is constant IBExtract.IntegralSubtypes.
LOG_default (see page 292)	This is constant IBExtract.LOG_default.
LOG_overflow (see page 292)	This is constant IBExtract.LOG_overflow.
LOG_raw (see page 293)	This is constant IBExtract.LOG_raw.
LOG_serial (see page 293)	This is constant IBExtract.LOG_serial.
MAX_INTSUBTYPES (see page 293)	This is constant IBExtract.MAX_INTSUBTYPES.
MAXSUBTYPES (see page 294)	This is constant IBExtract.MAXSUBTYPES.
obj_computed (see page 294)	This is constant IBExtract.obj_computed.
obj_count (see page 294)	This is constant IBExtract.obj_count.
obj_exception (see page 295)	This is constant IBExtract.obj_exception.
obj_expression_index (see page 295)	This is constant IBExtract.obj_expression_index.
obj_field (see page 295)	This is constant IBExtract.obj_field.
obj_index (see page 296)	This is constant IBExtract.obj_index.
obj_procedure (see page 296)	This is constant IBExtract.obj_procedure.
obj_relation (see page 296)	This is constant IBExtract.obj_relation.
obj_sql_role (see page 297)	This is constant IBExtract.obj_sql_role.
obj_trigger (see page 297)	This is constant IBExtract.obj_trigger.
obj_user (see page 297)	This is constant IBExtract.obj_user.
obj_user_group (see page 298)	This is constant IBExtract.obj_user_group.
obj_validation (see page 298)	This is constant IBExtract.obj_validation.
obj_view (see page 298)	This is constant IBExtract.obj_view.
ODS_VERSION10 (see page 299)	This is constant IBExtract.ODS_VERSION10.
ODS_VERSION6 (see page 299)	This is constant IBExtract.ODS_VERSION6.
ODS_VERSION7 (see page 299)	This is constant IBExtract.ODS_VERSION7.
ODS_VERSION8 (see page 300)	This is constant IBExtract.ODS_VERSION8.
ODS_VERSION9 (see page 300)	This is constant IBExtract.ODS_VERSION9.
priv_DELETE (see page 300)	This is constant IBExtract.priv_DELETE.
priv_EXECUTE (see page 301)	This is constant IBExtract.priv_EXECUTE.
priv_INSERT (see page 301)	This is constant IBExtract.priv_INSERT.
priv_REFERENCES (see page 301)	This is constant IBExtract.priv_REFERENCES.
priv_SELECT (see page 302)	This is constant IBExtract.priv_SELECT.
priv_UNKNOWN (see page 302)	This is constant IBExtract.priv_UNKNOWN.
priv_UPDATE (see page 302)	This is constant IBExtract.priv_UPDATE.
PrivTypes (see page 303)	This is constant IBExtract.PrivTypes.
SubTypes (see page 303)	This is constant IBExtract.SubTypes.
TriggerTypes (see page 303)	This is constant IBExtract.TriggerTypes.

8.4.1 IBExtract.ColumnTypes Constant

Pascal

```
ColumnTypes: TSQLTypes = ( (SqlType : blr_short; TypeName : 'SMALLINT'), (SqlType : blr_long;
TypeName : 'INTEGER'), (SqlType : blr_quad; TypeName : 'QUAD'), (SqlType : blr_float; TypeName
: 'FLOAT'), (SqlType : blr_text; TypeName : 'CHAR'), (SqlType : blr_double; TypeName : 'DOUBLE
PRECISION'), (SqlType : blr_varying; TypeName : 'VARCHAR'), (SqlType : blr_cstring; TypeName :
'CSTRING'), (SqlType : blr_blob_id; TypeName : 'BLOB_ID'), (SqlType : blr_blob; TypeName :
'BLOB'), (SqlType : blr_sql_time; TypeName : 'TIME'), (SqlType : blr_sql_date; TypeName :
'DATE'), (SqlType : blr_timestamp; TypeName : 'TIMESTAMP'), (SqlType : blr_int64; TypeName :
'INT64'));
```

C++

```
TSQLTypes ColumnTypes = ( (SqlType : blr_short; TypeName : 'SMALLINT'), (SqlType : blr_long;
TypeName : 'INTEGER'), (SqlType : blr_quad; TypeName : 'QUAD'), (SqlType : blr_float; TypeName
: 'FLOAT'), (SqlType : blr_text; TypeName : 'CHAR'), (SqlType : blr_double; TypeName : 'DOUBLE
PRECISION'), (SqlType : blr_varying; TypeName : 'VARCHAR'), (SqlType : blr_cstring; TypeName :
'CSTRING'), (SqlType : blr_blob_id; TypeName : 'BLOB_ID'), (SqlType : blr_blob; TypeName :
'BLOB'), (SqlType : blr_sql_time; TypeName : 'TIME'), (SqlType : blr_sql_date; TypeName :
'DATE'), (SqlType : blr_timestamp; TypeName : 'TIMESTAMP'), (SqlType : blr_int64; TypeName :
'INT64'));
```

File

IBExtract

Description

This is constant IBExtract.ColumnTypes.

8.4.2 IBExtract.FILE_cache Constant

Pascal

```
FILE_cache = 8;
```

C++

```
FILE_cache = 8;
```

File

IBExtract

Description

This is constant IBExtract.FILE_cache.

8.4.3 IBExtract.FILE_conditional Constant

Pascal

```
FILE_conditional = 16;
```

C++

```
FILE_conditional = 16;
```

File

IBExtract

Description

This is constant IBExtract.FILE_conditional.

8.4.4 IBExtract.FILE_inactive Constant

Pascal

```
FILE_inactive = 2;
```

C++

```
FILE_inactive = 2;
```

File

IBExtract

Description

This is constant IBExtract.FILE_inactive.

8.4.5 IBExtract.FILE_manual Constant

Pascal

```
FILE_manual = 4;
```

C++

```
FILE_manual = 4;
```

File

IBExtract

Description

This is constant IBExtract.FILE_manual.

8.4.6 IBExtract.FILE_shadow Constant

Pascal

```
FILE_shadow = 1;
```

C++

```
FILE_shadow = 1;
```

File

IBExtract

Description

This is constant IBExtract.FILE_shadow.

8.4.7 IBExtract.IntegralSubtypes Constant

Pascal

```
IntegralSubtypes: Array[0..2] of String = ( 'UNKNOWN', 'NUMERIC', 'DECIMAL');
```

C++

```
Array[0..2] of String IntegralSubtypes = ( 'UNKNOWN', 'NUMERIC', 'DECIMAL');
```

File

IBExtract

Description

This is constant IBExtract.IntegralSubtypes.

8.4.8 IBExtract.LOG_default Constant

Pascal

```
LOG_default = 2;
```

C++

```
LOG_default = 2;
```

File

IBExtract

Description

This is constant IBExtract.LOG_default.

8.4.9 IBExtract.LOG_overflow Constant

Pascal

```
LOG_overflow = 8;
```

C++

```
LOG_overflow = 8;
```

File

IBExtract

Description

This is constant IBExtract.LOG_overflow.

8.4.10 IBExtract.LOG_raw Constant

Pascal

```
LOG_raw = 4;
```

C++

```
LOG_raw = 4;
```

File

IBExtract

Description

This is constant IBExtract.LOG_raw.

8.4.11 IBExtract.LOG_serial Constant

Pascal

```
LOG_serial = 1;
```

C++

```
LOG_serial = 1;
```

File

IBExtract

Description

This is constant IBExtract.LOG_serial.

8.4.12 IBExtract.MAX_INTSUBTYPES Constant

Pascal

```
MAX_INTSUBTYPES = 2;
```

C++

```
MAX_INTSUBTYPES = 2;
```

File

IBExtract

Description

This is constant IBExtract.MAX_INTSUBTYPES.

8.4.13 IBExtract.MAXSUBTYPES Constant

Pascal

```
MAXSUBTYPES = 8;
```

C++

```
MAXSUBTYPES = 8;
```

File

IBExtract

Description

This is constant IBExtract.MAXSUBTYPES.

8.4.14 IBExtract.obj_computed Constant

Pascal

```
obj_computed = 3;
```

C++

```
obj_computed = 3;
```

File

IBExtract

Description

This is constant IBExtract.obj_computed.

8.4.15 IBExtract.obj_count Constant

Pascal

```
obj_count = 11;
```

C++

```
obj_count = 11;
```

File

IBExtract

Description

This is constant IBExtract.obj_count.

8.4.16 IBExtract.obj_exception Constant

Pascal

```
obj_exception = 7;
```

C++

```
obj_exception = 7;
```

File

IBExtract

Description

This is constant IBExtract.obj_exception.

8.4.17 IBExtract.obj_expression_index Constant

Pascal

```
obj_expression_index = 6;
```

C++

```
obj_expression_index = 6;
```

File

IBExtract

Description

This is constant IBExtract.obj_expression_index.

8.4.18 IBExtract.obj_field Constant

Pascal

```
obj_field = 9;
```

C++

```
obj_field = 9;
```

File

IBExtract

Description

This is constant IBExtract.obj_field.

8.4.19 IBExtract.obj_index Constant

Pascal

```
obj_index = 10;
```

C++

```
obj_index = 10;
```

File

IBExtract

Description

This is constant IBExtract.obj_index.

8.4.20 IBExtract.obj_procedure Constant

Pascal

```
obj_procedure = 5;
```

C++

```
obj_procedure = 5;
```

File

IBExtract

Description

This is constant IBExtract.obj_procedure.

8.4.21 IBExtract.obj_relation Constant

Pascal

```
obj_relation = 0;
```

C++

```
obj_relation = 0;
```

File

IBExtract

Description

This is constant IBExtract.obj_relation.

8.4.22 IBExtract.obj_sql_role Constant

Pascal

```
obj_sql_role = 13;
```

C++

```
obj_sql_role = 13;
```

File

IBExtract

Description

This is constant IBExtract.obj_sql_role.

8.4.23 IBExtract.obj_trigger Constant

Pascal

```
obj_trigger = 2;
```

C++

```
obj_trigger = 2;
```

File

IBExtract

Description

This is constant IBExtract.obj_trigger.

8.4.24 IBExtract.obj_user Constant

Pascal

```
obj_user = 8;
```

C++

```
obj_user = 8;
```

File

IBExtract

Description

This is constant IBExtract.obj_user.

8.4.25 IBExtract.obj_user_group Constant

Pascal

```
obj_user_group = 12;
```

C++

```
obj_user_group = 12;
```

File

IBExtract

Description

This is constant IBExtract.obj_user_group.

8.4.26 IBExtract.obj_validation Constant

Pascal

```
obj_validation = 4;
```

C++

```
obj_validation = 4;
```

File

IBExtract

Description

This is constant IBExtract.obj_validation.

8.4.27 IBExtract.obj_view Constant

Pascal

```
obj_view = 1;
```

C++

```
obj_view = 1;
```

File

IBExtract

Description

This is constant IBExtract.obj_view.

8.4.28 IBExtract.ODS_VERSION10 Constant

Pascal

```
ODS_VERSION10 = 10;
```

C++

```
ODS_VERSION10 = 10;
```

File

IBExtract

Description

This is constant IBExtract.ODS_VERSION10.

8.4.29 IBExtract.ODS_VERSION6 Constant

Pascal

```
ODS_VERSION6 = 6;
```

C++

```
ODS_VERSION6 = 6;
```

File

IBExtract

Description

This is constant IBExtract.ODS_VERSION6.

8.4.30 IBExtract.ODS_VERSION7 Constant

Pascal

```
ODS_VERSION7 = 7;
```

C++

```
ODS_VERSION7 = 7;
```

File

IBExtract

Description

This is constant IBExtract.ODS_VERSION7.

8.4.31 IBExtract.ODS_VERSION8 Constant

Pascal

```
ODS_VERSION8 = 8;
```

C++

```
ODS_VERSION8 = 8;
```

File

IBExtract

Description

This is constant IBExtract.ODS_VERSION8.

8.4.32 IBExtract.ODS_VERSION9 Constant

Pascal

```
ODS_VERSION9 = 9;
```

C++

```
ODS_VERSION9 = 9;
```

File

IBExtract

Description

This is constant IBExtract.ODS_VERSION9.

8.4.33 IBExtract.priv_DELETE Constant

Pascal

```
priv_DELETE = 16;
```

C++

```
priv_DELETE = 16;
```

File

IBExtract

Description

This is constant IBExtract.priv_DELETE.

8.4.34 IBExtract.priv_EXECUTE Constant

Pascal

```
priv_EXECUTE = 32;
```

C++

```
priv_EXECUTE = 32;
```

File

IBExtract

Description

This is constant IBExtract.priv_EXECUTE.

8.4.35 IBExtract.priv_INSERT Constant

Pascal

```
priv_INSERT = 4;
```

C++

```
priv_INSERT = 4;
```

File

IBExtract

Description

This is constant IBExtract.priv_INSERT.

8.4.36 IBExtract.priv_REFERENCES Constant

Pascal

```
priv_REFERENCES = 64;
```

C++

```
priv_REFERENCES = 64;
```

File

IBExtract

Description

This is constant IBExtract.priv_REFERENCES.

8.4.37 IBExtract.priv_SELECT Constant

Pascal

```
priv_SELECT = 2;
```

C++

```
priv_SELECT = 2;
```

File

IBExtract

Description

This is constant IBExtract.priv_SELECT.

8.4.38 IBExtract.priv_UNKNOWN Constant

Pascal

```
priv_UNKNOWN = 1;
```

C++

```
priv_UNKNOWN = 1;
```

File

IBExtract

Description

This is constant IBExtract.priv_UNKNOWN.

8.4.39 IBExtract.priv_UPDATE Constant

Pascal

```
priv_UPDATE = 8;
```

C++

```
priv_UPDATE = 8;
```

File

IBExtract

Description

This is constant IBExtract.priv_UPDATE.

8.4.40 IBExtract.PrivTypes Constant

Pascal

```
PrivTypes: Array[0..5] of TPrivTypes = ( (PrivFlag : priv_DELETE; PrivString : 'DELETE' ),
(PrivFlag : priv_EXECUTE; PrivString : 'EXECUTE' ), (PrivFlag : priv_INSERT; PrivString :
'INSERT' ), (PrivFlag : priv_SELECT; PrivString : 'SELECT' ), (PrivFlag : priv_UPDATE;
PrivString : 'UPDATE' ), (PrivFlag : priv_REFERENCES; PrivString : 'REFERENCES'));
```

C++

```
Array[0..5] of TPrivTypes PrivTypes = ( (PrivFlag : priv_DELETE; PrivString : 'DELETE' ),
(PrivFlag : priv_EXECUTE; PrivString : 'EXECUTE' ), (PrivFlag : priv_INSERT; PrivString :
'INSERT' ), (PrivFlag : priv_SELECT; PrivString : 'SELECT' ), (PrivFlag : priv_UPDATE;
PrivString : 'UPDATE' ), (PrivFlag : priv_REFERENCES; PrivString : 'REFERENCES'));
```

File

IBExtract

Description

This is constant IBExtract.PrivTypes.

8.4.41 IBExtract.SubTypes Constant

Pascal

```
SubTypes: Array[0..8] of String = ( 'UNKNOWN', 'TEXT', 'BLR', 'ACL', 'RANGES', 'SUMMARY',
'FORMAT', 'TRANSACTION_DESCRIPTION', 'EXTERNAL_FILE_DESCRIPTION');
```

C++

```
Array[0..8] of String SubTypes = ( 'UNKNOWN', 'TEXT', 'BLR', 'ACL', 'RANGES', 'SUMMARY',
'FORMAT', 'TRANSACTION_DESCRIPTION', 'EXTERNAL_FILE_DESCRIPTION');
```

File

IBExtract

Description

This is constant IBExtract.SubTypes.

8.4.42 IBExtract.TriggerTypes Constant

Pascal

```
TriggerTypes: Array[0..6] of String = ( '', 'BEFORE INSERT', 'AFTER INSERT', 'BEFORE UPDATE',
'AFTER UPDATE', 'BEFORE DELETE', 'AFTER DELETE');
```

C++

```
Array[0..6] of String TriggerTypes = ( '', 'BEFORE INSERT', 'AFTER INSERT', 'BEFORE UPDATE',
'AFTER UPDATE', 'BEFORE DELETE', 'AFTER DELETE');
```

File

IBExtract

Description

This is constant IBExtract.TriggerTypes.

9 IBInstall Namespace

This is namespace IBInstall.

Topics

Name	Description
Classes (see page 307)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 328)	The following table lists structs, records, enums in this documentation.
Types (see page 333)	The following table lists types in this documentation.

Classes

	Name	Description
	EIBInstall (see page 307)	EIBInstall catches all of the installation exceptions.
	EIBInstallerError (see page 309)	This is class IBInstall.EIBInstallerError.
	EIBInstallError (see page 310)	This is class IBInstall.EIBInstallError.
	TIBInstall (see page 310)	Use TIBInstall to set up an InterBase installation component.
	TIBSetup (see page 317)	TIBSetup is the ancestor object from which TIBInstall (see page 310) and TIBUnInstall (see page 322) are derived.
	TIBUnInstall (see page 322)	Use TIBUnInstall to set up an uninstall component.
	TInstallOptions (see page 325)	TInstallOptions configures the options of a TIBInstall (see page 310) instance.

Structs, Records, Enums

	Name	Description
	TCmdOption (see page 328)	TCmdOption and TCmdOptions (see page 333) indicate which of the command-line tools are installed.
	TConnectivityOption (see page 329)	TConnectivityOption and TConnectivityOptions (see page 333) identify InterBase connectivity clients.
	TErrorResult (see page 329)	This is record IBInstall.TErrorResult.
	TExamplesOption (see page 330)	TExamplesOption and TExamplesOptions (see page 334) identify InterBase Examples clients.
	TIBInstallerError (see page 330)	This is record IBInstall.TIBInstallerError.
	TMainOption (see page 331)	TMainOption and TMainOptions (see page 336) identify the main InterBase components.
	TStatusResult (see page 332)	This is record IBInstall.TStatusResult.
	TWarningResult (see page 332)	This is record IBInstall.TWarningResult.

Types

Name	Description
TCmdOptions (see page 333)	This is type IBInstall.TCmdOptions.
TConnectivityOptions (see page 333)	This is type IBInstall.TConnectivityOptions.
TExamplesOptions (see page 334)	This is type IBInstall.TExamplesOptions.
TIBSetupOnError (see page 334)	TIBSetupOnError is the type of event handlers that respond to installation errors.
TIBSetupOnStatus (see page 334)	TIBSetupOnStatus is the type of event handlers that respond during the InterBase installation or de-installation process.
TIBSetupOnWarning (see page 335)	TIBSetupOnWarning is the type of event handlers that respond to installation warnings.
TIscError (see page 335)	This is type IBInstall.TIscError.
TMainOptions (see page 336)	This is type IBInstall.TMainOptions.

9.1 Classes

The following table lists classes in this documentation.

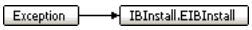
Classes

	Name	Description
	EIBInstall (see page 307)	EIBInstall catches all of the installation exceptions.
	EIBInstallerError (see page 309)	This is class IBInstall.EIBInstallerError.
	EIBInstallError (see page 310)	This is class IBInstall.EIBInstallError.
	TIBInstall (see page 310)	Use TIBInstall to set up an InterBase installation component.
	TIBSetup (see page 317)	TIBSetup is the ancestor object from which TIBInstall (see page 310) and TIBUnInstall (see page 322) are derived.
	TIBUnInstall (see page 322)	Use TIBUnInstall to set up an uninstall component.
	TInstallOptions (see page 325)	TInstallOptions configures the options of a TIBInstall (see page 310) instance.

9.1.1 EIBInstall Class

EIBInstall catches all of the installation exceptions.

Class Hierarchy



Pascal

```
EIBInstall = class(Exception);
```

C++

```
class EIBInstall : public Exception;
```

File

IBInstall

Description

Use EIBInstall to catch installation exceptions from InterBase and InterBase Express (IBX).

Note: You must install InterBase 6 to use this feature.

9.1.1.1 EIBInstall Members

The following tables list the members exposed by EIBInstall.

Public Methods

	Name	Description
	Create (see page 308)	This is the overview for the Create constructor overload.

Public Properties

	Name	Description
	InstallerError (see page 308)	Returns IBX error messages.

	InstallError (see page 309)	Returns InterBase installation error messages.

9.1.1.2 Create Constructor

This is the overview for the Create constructor overload.

Overload List

	Name	Description
	EIBInstall.Create (MSG_NO, string) (see page 308)	Creates an instance of an EIBInstall (see page 307) object.
	EIBInstall.Create (TIBInstallerError, string) (see page 308)	Creates an instance of an EIBInstall (see page 307) object.

9.1.1.2.1 EIBInstall.Create Constructor (MSG_NO, string)

Creates an instance of an EIBInstall (see page 307) object.

Pascal

```
constructor Create(IscCode: MSG_NO; IscMessage: string); virtual; overload;
constructor Create(ECode: TIBInstallerError; EMessage: string); virtual; overload;
```

C++

```
virtual __fastcall EIBInstall(MSG_NO IscCode, UnicodeString IscMessage);
virtual __fastcall EIBInstall(TIBInstallerError ECode, UnicodeString EMessage);
```

Description

Call Create to create an instance of an EIBInstall (see page 307) exception object.

Note: You must install InterBase 6 to use this feature.

9.1.1.3 EIBInstall Properties

The properties of the EIBInstall class are listed here.

Public Properties

	Name	Description
	InstallerError (see page 308)	Returns IBX error messages.
	InstallError (see page 309)	Returns InterBase installation error messages.

9.1.1.3.1 EIBInstall.InstallerError Property

Returns IBX error messages.

Pascal

```
property InstallerError: TIBInstallerError;
```

C++

```
__property TIBInstallerError InstallerError;
```

Description

Use InstallerError to return error messages generated by the IBX install components. InstallerError can be one of:

ieSuccess	Installation is successful
ieDelphiException	Installation fails due to a Delphi exception
ieNoOptionsSet	Installation fails because no installation options are set
ieNoDestinationDirectory	Installation fails because no destination directory is set
ieNoSourceDirectory	Installation fails because no source directory is set
ieNoUninstallFile	Installation fails because no uninstall file is named
ieOptionNeedsClient	Installation fails because the installation needs the InterBase Client component to succeed
ieOptionNeedsServer	Installation fails because the installation needs the InterBase Server component to succeed
ieInvalidOption	Option is invalid
ieInvalidOnErrorResult	The OnError event result is invalid
ieInvalidOnStatusResult	The OnStatus event result is invalid

Note: You must install InterBase 6 to use this feature.

9.1.1.3.2 EIBInstall.InstallError Property

Returns InterBase installation error messages.

Pascal

```
property InstallError: MSG_NO;
```

C++

```
__property MSG_NO InstallError;
```

Description

Use InstallError to return error messages generated by the InterBase API.

Note: You must install InterBase 6 to use this feature.

9.1.2 EIBInstallerError Class

Class Hierarchy

Pascal

```
EIBInstallerError = class(EIBInstall);
```

C++

```
class EIBInstallerError : public EIBInstall;
```

File

IBInstall

Description

This is class IBInstall.EIBInstallerError.

9.1.3 EIBInstallError Class

Class Hierarchy

Pascal

```
EIBInstallError = class(EIBInstall);
```

C++

```
class EIBInstallError : public EIBInstall;
```

File

IBInstall

Description

This is class IBInstall.EIBInstallError.

9.1.4 TIBInstall Class

Use TIBInstall to set up an InterBase installation component.

Class Hierarchy

Pascal

```
TIBInstall = class(TIBSetup);
```

C++

```
class TIBInstall : public TIBSetup;
```

File

IBInstall

Description

Use TIBInstall to set up an InterBase installation component.

Note: To use this component, you must install InterBase 6 and you must place ibinstall.dll in the path.

See Also

Getting Started with InterBase Express

9.1.4.1 TIBInstall Members

The following tables list the members exposed by TIBInstall.

Protected Methods

	Name	Description
	Call (see page 318)	This is Call, a member of class TIBSetup.
	ErrorInternal (see page 318)	This is ErrorInternal, a member of class TIBSetup.
	GetInstallMessage (see page 319)	This is GetInstallMessage, a member of class TIBSetup.
	IBInstallError (see page 319)	This is IBInstallError, a member of class TIBSetup.
	StatusInternal (see page 319)	This is StatusInternal, a member of class TIBSetup.

Public Methods

	Name	Description
V	Create (see page 311)	Creates an installation component.
V	Destroy (see page 312)	Destroys the instance of TIBInstall (see page 310).
	GetOptionDescription (see page 312)	This is the overview for the GetOptionDescription method overload.
	GetOptionName (see page 313)	This is the overview for the GetOptionName method overload.
	GetOptionSpaceRequired (see page 313)	This is the overview for the GetOptionSpaceRequired method overload.
	InstallCheck (see page 314)	Checks if the installation is ready to be performed.
	InstallExecute (see page 314)	Executes the installation.

Public Properties

	Name	Description
	ErrorContext (see page 319)	Sets or returns the error context.
	MsgFilePath (see page 320)	Sets or returns the installation message file path.
R	Progress (see page 320)	Returns the installation progress.
R	RebootToComplete (see page 320)	Sets whether it is necessary to reboot to complete the installation.
	StatusContext (see page 321)	Sets or returns the status context.
R	SuggestedDestination (see page 316)	Returns the suggested installation target destination.
R	UnInstallFile (see page 316)	Returns the name and path of the uninstall file.

Published Events

	Name	Description
	OnError (see page 321)	Occurs when an installation error occurs.
	OnStatusChange (see page 322)	Occurs when a change in installation status occurs.
	OnWarning (see page 322)	Occurs when an installation warning occurs.

Published Properties

	Name	Description
	DestinationDirectory (see page 315)	Sets or returns the installation target path.
	InstallOptions (see page 315)	Sets or displays the installation options.
	SourceDirectory (see page 316)	Sets or returns the path of the InterBase installation source files.

9.1.4.2 TIBInstall.Create Constructor

Creates an installation component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBInstall(TComponent * AOwner);
```

Description

Call (see page 318) Create to create an installation component.

Note: You must install InterBase 6 to use this feature.

9.1.4.3 TIBInstall.Destroy Destructor

Destroys the instance of TIBInstall (see page 310).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBInstall();
```

Description

Do not call Destroy directly in an application. Instead, an application should call Free. Free verifies that the install object has not already been freed before it calls Destroy.

Note: You must install InterBase 6 to use this feature.

9.1.4.4 TIBInstall Methods

The methods of the TIBInstall class are listed here.

Public Methods

	Name	Description
	GetOptionDescription (see page 312)	This is the overview for the GetOptionDescription method overload.
	GetOptionName (see page 313)	This is the overview for the GetOptionName method overload.
	GetOptionSpaceRequired (see page 313)	This is the overview for the GetOptionSpaceRequired method overload.
	InstallCheck (see page 314)	Checks if the installation is ready to be performed.
	InstallExecute (see page 314)	Executes the installation.

9.1.4.4.1 GetOptionDescription Method

This is the overview for the GetOptionDescription method overload.

Overload List

	Name	Description
	TIBInstall.GetOptionDescription (TCmdOption) (see page 312)	Returns a description of a specified option.
	TIBInstall.GetOptionDescription (TConnectivityOption) (see page 312)	Returns a description of a specified option.
	TIBInstall.GetOptionDescription (TExamplesOption) (see page 312)	Returns a description of a specified option.
	TIBInstall.GetOptionDescription (TMainOption) (see page 312)	Returns a description of a specified option.

9.1.4.4.1.1 TIBInstall.GetOptionDescription Method (TCmdOption)

Returns a description of a specified option.

Pascal

```
function GetOptionDescription(Option: TExamplesOption): string; overload;
function GetOptionDescription(Option: TMainOption): string; overload;
```

```
function GetOptionDescription(Option: TConnectivityOption): string; overload;
function GetOptionDescription(Option: TCmdOption): string; overload;
```

C++

```
__fastcall UnicodeString GetOptionDescription(TExamplesOption Option);
__fastcall UnicodeString GetOptionDescription(TMainOption Option);
__fastcall UnicodeString GetOptionDescription(TConnectivityOption Option);
__fastcall UnicodeString GetOptionDescription(TCmdOption Option);
```

Description

Call (see page 318) GetOptionDescription to return a description of the indicated option. If the specified option is not valid, an error is returned.

Note: You must install InterBase 6 to use this feature.

9.1.4.4.2 GetOptionName Method

This is the overview for the GetOptionName method overload.

Overload List

	Name	Description
	TIBInstall.GetOptionName (TCmdOption) (see page 313)	Returns the option name.
	TIBInstall.GetOptionName (TConnectivityOption) (see page 313)	Returns the option name.
	TIBInstall.GetOptionName (TExamplesOption) (see page 313)	Returns the option name.
	TIBInstall.GetOptionName (TMainOption) (see page 313)	Returns the option name.

9.1.4.4.2.1 TIBInstall.GetOptionName Method (TCmdOption)

Returns the option name.

Pascal

```
function GetOptionName(Option: TExamplesOption): string; overload;
function GetOptionName(Option: TMainOption): string; overload;
function GetOptionName(Option: TConnectivityOption): string; overload;
function GetOptionName(Option: TCmdOption): string; overload;
```

C++

```
__fastcall UnicodeString GetOptionName(TExamplesOption Option);
__fastcall UnicodeString GetOptionName(TMainOption Option);
__fastcall UnicodeString GetOptionName(TConnectivityOption Option);
__fastcall UnicodeString GetOptionName(TCmdOption Option);
```

Description

Call (see page 318) GetOptionName to return the name of the indicated option.

Note: You must install InterBase 6 to use this feature.

9.1.4.4.3 GetOptionSpaceRequired Method

This is the overview for the GetOptionSpaceRequired method overload.

Overload List

	Name	Description
	TIBInstall.GetOptionSpaceRequired (TCmdOption) (see page 314)	Returns the space required to install the indicated option.
	TIBInstall.GetOptionSpaceRequired (TConnectivityOption) (see page 314)	Returns the space required to install the indicated option.
	TIBInstall.GetOptionSpaceRequired (TExamplesOption) (see page 314)	Returns the space required to install the indicated option.
	TIBInstall.GetOptionSpaceRequired (TMainOption) (see page 314)	Returns the space required to install the indicated option.

9.1.4.4.3.1 TIBInstall.GetOptionSpaceRequired Method (TCmdOption)

Returns the space required to install the indicated option.

Pascal

```
function GetOptionSpaceRequired(Option: TExamplesOption): longword; overload;
function GetOptionSpaceRequired(Option: TMainOption): longword; overload;
function GetOptionSpaceRequired(Option: TConnectivityOption): longword; overload;
function GetOptionSpaceRequired(Option: TCmdOption): longword; overload;
```

C++

```
__fastcall longword GetOptionSpaceRequired(TExamplesOption Option);
__fastcall longword GetOptionSpaceRequired(TMainOption Option);
__fastcall longword GetOptionSpaceRequired(TConnectivityOption Option);
__fastcall longword GetOptionSpaceRequired(TCmdOption Option);
```

Description

Call (see page 318) GetOptionSpaceRequired to return the space required in kilobytes for the indicated option.

Note: You must install InterBase 6 to use this feature.

9.1.4.4.4 TIBInstall.InstallCheck Method

Checks if the installation is ready to be performed.

Pascal

```
procedure InstallCheck;
```

C++

```
__fastcall InstallCheck();
```

Description

Call (see page 318) InstallCheck to check if everything is ready for the installation to be performed.

Note: You must install InterBase 6 to use this feature.

9.1.4.4.5 TIBInstall.InstallExecute Method

Executes the installation.

Pascal

```
procedure InstallExecute;
```

C++

```
__fastcall InstallExecute();
```

Description

Call (see page 318) InstallExecute to execute the installation.

Note: You must install InterBase 6 to use this feature.

9.1.4.5 TIBInstall Properties

The properties of the TIBInstall class are listed here.

Public Properties

	Name	Description
	SuggestedDestination (see page 316)	Returns the suggested installation target destination.
	UnInstallFile (see page 316)	Returns the name and path of the uninstall file.

Published Properties

	Name	Description
	DestinationDirectory (see page 315)	Sets or returns the installation target path.
	InstallOptions (see page 315)	Sets or displays the installation options.
	SourceDirectory (see page 316)	Sets or returns the path of the InterBase installation source files.

9.1.4.5.1 TIBInstall.DestinationDirectory Property

Sets or returns the installation target path.

Pascal

```
property DestinationDirectory: string;
```

C++

```
__property UnicodeString DestinationDirectory;
```

Description

Use DestinationDirectory to set or returns the installation target path. Initially, this will be the same as the SuggestedDestination (see page 316).

Note: You must install InterBase 6 to use this feature.

9.1.4.5.2 TIBInstall.InstallOptions Property

Sets or displays the installation options.

Pascal

```
property InstallOptions: TInstallOptions;
```

C++

```
__property TInstallOptions InstallOptions;
```

Description

Use InstallOptions to set or display the installation options (TInstallOptions (see page 325)). TInstallOptions (see page 325) are:

Option	Installs:
CmdLineTools	The InterBase command line tools
ConnectivityClients	The InterBase connectivity clients
Examples	The InterBase examples
MainComponents	The main InterBase components

Note: You must install InterBase 6 to use this feature.

9.1.4.5.3 TIBInstall.SourceDirectory Property

Sets or returns the path of the InterBase installation source files.

Pascal

```
property SourceDirectory: string;
```

C++

```
__property UnicodeString SourceDirectory;
```

Description

Use SourceDirectory to set or return the path of the installation source files. In most cases, this will be a path on the InterBase CDROM.

Note: You must install InterBase 6 to use this feature.

9.1.4.5.4 TIBInstall.SuggestedDestination Property

Returns the suggested installation target destination.

Pascal

```
property SuggestedDestination: string;
```

C++

```
__property UnicodeString SuggestedDestination;
```

Description

Use SuggestedDestination to return suggested installation target destination.

Note: You must install InterBase 6 to use this feature.

9.1.4.5.5 TIBInstall.UnInstallFile Property

Returns the name and path of the uninstall file.

Pascal

```
property UnInstallFile: string;
```

C++

```
__property UnicodeString UnInstallFile;
```

Description

Use UnInstallFile to return the binary log file name and path of the uninstall file. The uninstall file contains information on which options were installed. This information lets the TIBUnInstall (see page 322) component know what needs to be removed.

Note: You must install InterBase 6 to use this feature.

9.1.5 TIBSetup Class

TIBSetup is the ancestor object from which TIBInstall (see page 310) and TIBUnInstall (see page 322) are derived.

Class Hierarchy

Pascal

```
TIBSetup = class(TComponent);
```

C++

```
class TIBSetup : public TComponent;
```

File

IBInstall

Description

TIBSetup is the ancestor object from which TIBInstall (see page 310) and TIBUnInstall (see page 322) are derived.

Note: You must install InterBase 6 to use this feature.

9.1.5.1 TIBSetup Members

The following tables list the members exposed by TIBSetup.

Protected Methods

	Name	Description
	Call (see page 318)	This is Call, a member of class TIBSetup.
	ErrorInternal (see page 318)	This is ErrorInternal, a member of class TIBSetup.
	GetInstallMessage (see page 319)	This is GetInstallMessage, a member of class TIBSetup.
	IBInstallError (see page 319)	This is IBInstallError, a member of class TIBSetup.
	StatusInternal (see page 319)	This is StatusInternal, a member of class TIBSetup.

Public Methods

	Name	Description
V	Create (see page 318)	Creates an installation component.

Public Properties

	Name	Description
	ErrorContext (see page 319)	Sets or returns the error context.
	MsgFilePath (see page 320)	Sets or returns the installation message file path.
R	Progress (see page 320)	Returns the installation progress.
R	RebootToComplete (see page 320)	Sets whether it is necessary to reboot to complete the installation.
	StatusContext (see page 321)	Sets or returns the status context.

Published Events

	Name	Description
	OnError (see page 321)	Occurs when an installation error occurs.
	OnStatusChange (see page 322)	Occurs when a change in installation status occurs.

	OnWarning (see page 322)	Occurs when an installation warning occurs.

9.1.5.2 TIBSetup.Create Constructor

Creates an installation component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBSetup(TComponent * AOwner);
```

Description

Call (see page 318) Create to create an installation component.

Note: You must install InterBase 6 to use this feature.

9.1.5.3 TIBSetup Methods

The methods of the TIBSetup class are listed here.

Protected Methods

	Name	Description
	Call (see page 318)	This is Call, a member of class TIBSetup.
	ErrorInternal (see page 318)	This is ErrorInternal, a member of class TIBSetup.
	GetInstallMessage (see page 319)	This is GetInstallMessage, a member of class TIBSetup.
	IBInstallError (see page 319)	This is IBInstallError, a member of class TIBSetup.
	StatusInternal (see page 319)	This is StatusInternal, a member of class TIBSetup.

9.1.5.3.1 TIBSetup.Call Method

Pascal

```
procedure Call(IscCode: MSG_NO);
```

C++

```
__fastcall Call(MSG_NO IscCode);
```

Description

This is Call, a member of class TIBSetup.

9.1.5.3.2 TIBSetup.ErrorInternal Method

Pascal

```
function ErrorInternal(IscCode: MSG_NO; const ActionDescription: TEXT): Integer;
```

C++

```
__fastcall int ErrorInternal(MSG_NO IscCode, const TEXT ActionDescription);
```

Description

This is ErrorInternal, a member of class TIBSetup.

9.1.5.3.3 TIBSetup.GetInstallMessage Method

Pascal

```
function GetInstallMessage(IscCode: MSG_NO): string;
```

C++

```
__fastcall UnicodeString GetInstallMessage(MSG_NO IscCode);
```

Description

This is GetInstallMessage, a member of class TIBSetup.

9.1.5.3.4 TIBSetup.IBInstallError Method

Pascal

```
procedure IBInstallError(IscCode: MSG_NO);
```

C++

```
__fastcall IBInstallError(MSG_NO IscCode);
```

Description

This is IBInstallError, a member of class TIBSetup.

9.1.5.3.5 TIBSetup.StatusInternal Method

Pascal

```
function StatusInternal(Status: Integer; const ActionDescription: TEXT): Integer;
```

C++

```
__fastcall int StatusInternal(int Status, const TEXT ActionDescription);
```

Description

This is StatusInternal, a member of class TIBSetup.

9.1.5.4 TIBSetup Properties

The properties of the TIBSetup class are listed here.

Public Properties

	Name	Description
	ErrorContext (see page 319)	Sets or returns the error context.
	MsgFilePath (see page 320)	Sets or returns the installation message file path.
R	Progress (see page 320)	Returns the installation progress.
R	RebootToComplete (see page 320)	Sets whether it is necessary to reboot to complete the installation.
	StatusContext (see page 321)	Sets or returns the status context.

9.1.5.4.1 TIBSetup.ErrorContext Property

Sets or returns the error context.

Pascal

```
property ErrorContext: Pointer;
```

C++

```
__property void * ErrorContext;
```

Description

Use ErrorContext to set or return the user-defined data for the error function.

Note: You must install InterBase 6 to use this feature.

9.1.5.4.2 TIBSetup.MsgFilePath Property

Sets or returns the installation message file path.

Pascal

```
property MsgFilePath: string;
```

C++

```
__property UnicodeString MsgFilePath;
```

Description

Use MsgFilePath to set or return the full directory path to the filename.msg file. Most of the time this is the same as the source and path where the InterBase product is installed.

This property allows you to supply your own installation messages. It is also possible to write the setup program in such a way that you could switch languages on demand, by providing several filename.msg files at different locations.

The filename.msg file normally contains:

- Error messages
- Option names and descriptions
- Actions descriptions and status messages that are logged to the ibinstall.log file

Note: You must install InterBase 6 to use this feature.

9.1.5.4.3 TIBSetup.Progress Property

Returns the installation progress.

Pascal

```
property Progress: Integer;
```

C++

```
__property int Progress;
```

Description

Use Progress to return the installation progress as an integer between 0 and 100. Progress is updated by the OnStatusChange (see page 322) event.

Note: You must install InterBase 6 to use this feature.

9.1.5.4.4 TIBSetup.RebootToComplete Property

Sets whether it is necessary to reboot to complete the installation.

Pascal

```
property RebootToComplete: Boolean;
```

C++

```
__property Boolean RebootToComplete;
```

Description

Use RebootToComplete to set whether it is necessary to reboot the system to complete the installation or uninstallation. By default, this is not necessary (False).

Note: You must install InterBase 6 to use this feature.

9.1.5.4.5 TIBSetup.StatusContext Property

Sets or returns the status context.

Pascal

```
property StatusContext: Pointer;
```

C++

```
__property void * StatusContext;
```

Description

Use StatusContext to set or return the user-defined data for the status function.

Note: You must install InterBase 6 to use this feature.

9.1.5.5 TIBSetup Events

The events of the TIBSetup class are listed here.

Published Events

	Name	Description
	OnError (see page 321)	Occurs when an installation error occurs.
	OnStatusChange (see page 322)	Occurs when a change in installation status occurs.
	OnWarning (see page 322)	Occurs when an installation warning occurs.

9.1.5.5.1 TIBSetup.OnError Event

Occurs when an installation error occurs.

Pascal

```
property OnError: TIBSetupOnError;
```

C++

```
__property TIBSetupOnError OnError;
```

Description

Write an OnError event handler to take specific actions when an error occurs during installation.

Note: You must install InterBase 6 to use this feature.

9.1.5.5.2 TIBSetup.OnStatusChange Event

Occurs when a change in installation status occurs.

Pascal

```
property OnStatusChange: TIBSetupOnStatus;
```

C++

```
__property TIBSetupOnStatus OnStatusChange;
```

Description

Write an OnStatusChange event handler to take specific actions when a status change occurs during installation.

Status events are generated from time to time when InstallExecute is called. The event informs the user of the installation progress. The event handler might want to use the StatusContext (see page 321) property to display information about the current installation status. The progress is returned by the Progress (see page 320) property as an integer from 1-100.

Note: You must install InterBase 6 to use this feature.

9.1.5.5.3 TIBSetup.OnWarning Event

Occurs when an installation warning occurs.

Pascal

```
property OnWarning: TIBSetupOnWarning;
```

C++

```
__property TIBSetupOnWarning OnWarning;
```

Description

Write an OnWarning event handler to take specific actions when a warning occurs.

An OnWarning event is generated whenever warning is reported, and must return either isc_fp_install_continue or isc_install_abort.

Note: You must install InterBase 6 to use this feature.

9.1.6 TIBUnInstall Class

Use TIBUnInstall to set up an uninstall component.

Class Hierarchy

Pascal

```
TIBUnInstall = class(TIBSetup);
```

C++

```
class TIBUnInstall : public TIBSetup;
```

File

IBInstall

Description

Use TIBUnInstall to set up an uninstall component.

Note: You must install InterBase 6 to use this feature.

See Also

Getting Started with InterBase Express

9.1.6.1 TIBUnInstall Members

The following tables list the members exposed by TIBUnInstall.

Protected Methods

	Name	Description
	Call (see page 318)	This is Call, a member of class TIBSetup.
	ErrorInternal (see page 318)	This is ErrorInternal, a member of class TIBSetup.
	GetInstallMessage (see page 319)	This is GetInstallMessage, a member of class TIBSetup.
	IBInstallError (see page 319)	This is IBInstallError, a member of class TIBSetup.
	StatusInternal (see page 319)	This is StatusInternal, a member of class TIBSetup.

Public Methods

	Name	Description
V	Create (see page 318)	Creates an installation component.
	UnInstallCheck (see page 324)	Checks to see if uninstallation is possible.
	UnInstallExecute (see page 324)	Executes the uninstallation.

Public Properties

	Name	Description
	ErrorContext (see page 319)	Sets or returns the error context.
	MsgFilePath (see page 320)	Sets or returns the installation message file path.
R	Progress (see page 320)	Returns the installation progress.
R	RebootToComplete (see page 320)	Sets whether it is necessary to reboot to complete the installation.
	StatusContext (see page 321)	Sets or returns the status context.
	UnInstallFile (see page 324)	Returns the name and path of the InterBase uninstall file.

Published Events

	Name	Description
	OnError (see page 321)	Occurs when an installation error occurs.
	OnStatusChange (see page 322)	Occurs when a change in installation status occurs.
	OnWarning (see page 322)	Occurs when an installation warning occurs.

9.1.6.2 TIBUnInstall Methods

The methods of the TIBUnInstall class are listed here.

Public Methods

	Name	Description
	UnInstallCheck (see page 324)	Checks to see if uninstallation is possible.
	UnInstallExecute (see page 324)	Executes the uninstallation.

9.1.6.2.1 TIBUnInstall.UnInstallCheck Method

Checks to see if uninstallation is possible.

Pascal

```
procedure UnInstallCheck;
```

C++

```
__fastcall UnInstallCheck();
```

Description

Call (see page 318) UnInstallCheck to check if uninstallation is possible.

Note: You must install InterBase 6 to use this feature.

9.1.6.2.2 TIBUnInstall.UnInstallExecute Method

Executes the uninstallation.

Pascal

```
procedure UnInstallExecute;
```

C++

```
__fastcall UnInstallExecute();
```

Description

Call (see page 318) UnInstallExecute to execute the uninstallation.

Note: You must install InterBase 6 to use this feature.

9.1.6.3 TIBUnInstall Properties

The properties of the TIBUnInstall class are listed here.

Public Properties

	Name	Description
	UnInstallFile (see page 324)	Returns the name and path of the InterBase uninstall file.

9.1.6.3.1 TIBUnInstall.UnInstallFile Property

Returns the name and path of the InterBase uninstall file.

Pascal

```
property UnInstallFile: string;
```

C++

```
__property UnicodeString UnInstallFile;
```

Description

Use UnInstallFile to return the name and path of the InterBase uninstall file.

Note: You must install InterBase 6 to use this feature.

9.1.7 TInstallOptions Class

TInstallOptions configures the options of a TIBInstall (see page 310) instance.

Class Hierarchy

Pascal

```
TInstallOptions = class(TPersistent);
```

C++

```
class TInstallOptions : public TPersistent;
```

File

IBInstall

Description

Use TInstallOptions to indicate the options to use when installing InterBase. This is the type of the Options property on TIBInstall (see page 310).

Note: You must install InterBase 6 to use this feature.

9.1.7.1 TInstallOptions Members

The following tables list the members exposed by TInstallOptions.

Published Properties

	Name	Description
	CmdLineTools (see page 325)	Installs the command line tools for InterBase.
	ConnectivityClients (see page 326)	Installs the connectivity clients for InterBase.
	Examples (see page 326)	Installs all the InterBase examples.
	MainComponents (see page 326)	Installs the InterBase main components.

9.1.7.2 TInstallOptions Properties

The properties of the TInstallOptions class are listed here.

Published Properties

	Name	Description
	CmdLineTools (see page 325)	Installs the command line tools for InterBase.
	ConnectivityClients (see page 326)	Installs the connectivity clients for InterBase.
	Examples (see page 326)	Installs all the InterBase examples.
	MainComponents (see page 326)	Installs the InterBase main components.

9.1.7.2.1 TInstallOptions.CmdLineTools Property

Installs the command line tools for InterBase.

Pascal

```
property CmdLineTools: TCmdOptions;
```

C++

```
__property TCmdOptions CmdLineTools;
```

Description

Use CmdLineTools to indicate which of the command line tools for InterBase should be installed. The installer issues a warning for any of these if the MainComponents (see page 326) property does not include moClient.

Note: You must install InterBase 6 to use this feature.

9.1.7.2.2 TInstallOptions.ConnectivityClients Property

Installs the connectivity clients for InterBase.

Pascal

```
property ConnectivityClients: TConnectivityOptions;
```

C++

```
__property TConnectivityOptions ConnectivityClients;
```

Description

Use ConnectivityClients to install the InterBase connectivity clients.

Note: You must install InterBase 6 to use this feature.

9.1.7.2.3 TInstallOptions.Examples Property

Installs all the InterBase examples.

Pascal

```
property Examples: TExamplesOptions;
```

C++

```
__property TExamplesOptions Examples;
```

Description

Use Examples to indicates which of the InterBase examples should be installed.

When Examples includes exAPI, the MainComponents (see page 326) property must include moClient and moDevelopment. When Examples includes exDev, MainComponents (see page 326) must include moServer.

Note: You must install InterBase 6 to use this feature.

9.1.7.2.4 TInstallOptions.MainComponents Property

Installs the InterBase main components.

Pascal

```
property MainComponents: TMainOptions;
```

C++

```
__property TMainOptions MainComponents;
```

Description

Use MainComponents to indicate which of the InterBase main components to install. These can include the client, server, development tools, documentation, and ODBC driver and driver manager.

Note: You must install InterBase 6 to use this feature.

9.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TCmdOption (see page 328)	TCmdOption and TCmdOptions (see page 333) indicate which of the command-line tools are installed.
	TConnectivityOption (see page 329)	TConnectivityOption and TConnectivityOptions (see page 333) identify InterBase connectivity clients.
	TErrorResult (see page 329)	This is record IBInstall.TErrorResult.
	TExamplesOption (see page 330)	TExamplesOption and TExamplesOptions (see page 334) identify InterBase Examples clients.
	TIBInstallerError (see page 330)	This is record IBInstall.TIBInstallerError.
	TMainOption (see page 331)	TMainOption and TMainOptions (see page 336) identify the main InterBase components.
	TStatusResult (see page 332)	This is record IBInstall.TStatusResult.
	TWarningResult (see page 332)	This is record IBInstall.TWarningResult.

9.2.1 IBInstall.TCmdOption Enumeration

TCmdOption and TCmdOptions (see page 333) indicate which of the command-line tools are installed.

Pascal

```
TCmdOption = (
  cmDBMgmt,
  cmDBQuery,
  cmUsrMgmt
);
```

C++

```
enum TCmdOption {
  cmDBMgmt,
  cmDBQuery,
  cmUsrMgmt
};
```

File

IBInstall

Description

TCmdOption indicates a particular command-line tool for InterBase. TCmdOptions (see page 333) is a set of TCmdOption values that indicates the tools that are installed. The following table lists the TCmdOption values:

Value Meaning

cmDBMgmt gbak and gfix database management tools.

cmDBQuery isql, gdef, and qli database query tools.

cmUsrMgmt gsec user management tool.

9.2.2 IBInstall.TConnectivityOption Enumeration

TConnectivityOption and TConnectivityOptions (see page 333) identify InterBase connectivity clients.

Pascal

```
TConnectivityOption = (
  cnODBC,
  cnOLEDB,
  cnJDBC
);
```

C++

```
enum TConnectivityOption {
  cnODBC,
  cnOLEDB,
  cnJDBC
};
```

File

IBInstall

Description

TConnectivityOption indicates a particular connectivity client. TConnectivityOptions (see page 333) is a set of TConnectivityOption values that indicates the clients that are installed. The following table lists the TConnectivityOption values:

Value Meaning

cnODBC The Open Database Connectivity client.

cnOLEDB The OLE DB connectivity client.

cnJDBC The InterClient Java (see page 987) connectivity client.

9.2.3 IBInstall.TErrorResult Enumeration

Pascal

```
TErrorResult = (
  erAbort,
  erContinue,
  erRetry
);
```

C++

```
enum TErrorResult {
  erAbort,
  erContinue,
  erRetry
};
```

File

IBInstall

Description

This is record IBInstall.TErrorResult.

9.2.4 IBInstall.TExamplesOption Enumeration

TExamplesOption and TExamplesOptions (see page 334) identify InterBase Examples clients.

Pascal

```
TExamplesOption = (
  exDB,
  exAPI
);
```

C++

```
enum TExamplesOption {
  exDB,
  exAPI
};
```

File

IBInstall

Description

TExamplesOption indicates a particular example that ships with InterBase. TExamplesOptions (see page 334) is a set of TExamplesOption values that indicates the examples that are installed. The following table lists the TExamplesOption values:

Value Meaning

exAPI API, SQL, DSQL, and ESQL example files.

exDB Database examples.

9.2.5 IBInstall.TIBInstallerError Enumeration

Pascal

```
TIBInstallerError = (
  ieSuccess,
  ieDelphiException,
  ieNoOptionsSet,
  ieNoDestinationDirectory,
  ieNosourceDirectory,
  ieNoUninstallFile,
  ieOptionNeedsClient,
  ieOptionNeedsServer,
  ieInvalidOption,
  ieInvalidOnErrorResult,
  ieInvalidOnStatusResult
);
```

C++

```
enum TIBInstallerError {
  ieSuccess,
  ieDelphiException,
```

```
    ieNoOptionsSet,
    ieNoDestinationDirectory,
    ieNosourceDirectory,
    ieNoUninstallFile,
    ieOptionNeedsClient,
    ieOptionNeedsServer,
    ieInvalidOption,
    ieInvalidOnErrorResult,
    ieInvalidOnStatusResult
  };
```

File

IBInstall

Description

This is record IBInstall.TIBInstallerError.

9.2.6 IBInstall.TMainOption Enumeration

TMainOption and TMainOptions (see page 336) identify the main InterBase components.

Pascal

```
  TMainOption = (
    moServer,
    moClient,
    moConServer,
    moGuiTools,
    moDocumentation,
    moDevelopment
  );
```

C++

```
  enum TMainOption {
    moServer,
    moClient,
    moConServer,
    moGuiTools,
    moDocumentation,
    moDevelopment
  };
```

File

IBInstall

Description

TMainOption indicates one of the main InterBase components. TMainOptions (see page 336) is a set of TMainOption values that indicates the components that are installed. The following table lists the TMainOption values:

Value Meaning

moServer Server component, including: the server, license file (if present), message file, the Guardian, server configuration tool, the database information tools, UDF library, International character set library, and help files (Windows only).

moClient Client component, including the client library, license file, and message file.

moConServer Connectivity server, for future implementation.

moGuiTools Graphical interface tools, including IBConsole.

moDocumentation Documentation, including PDF files and indexes.

moDevelopment Development tools, including gpre, the import libraries, and the header files.

9.2.7 IBInstall.TStatusResult Enumeration

Pascal

```
TStatusResult = (
  srAbort,
  srContinue
);
```

C++

```
enum TStatusResult {
  srAbort,
  srContinue
};
```

File

IBInstall

Description

This is record IBInstall.TStatusResult.

9.2.8 IBInstall.TWarningResult Enumeration

Pascal

```
TWarningResult = (
  wrAbort,
  wrContinue
);
```

C++

```
enum TWarningResult {
  wrAbort,
  wrContinue
};
```

File

IBInstall

Description

This is record IBInstall.TWarningResult.

9.3 Types

The following table lists types in this documentation.

Types

Name	Description
TCmdOptions (see page 333)	This is type IBInstall.TCmdOptions.
TConnectivityOptions (see page 333)	This is type IBInstall.TConnectivityOptions.
TExamplesOptions (see page 334)	This is type IBInstall.TExamplesOptions.
TIBSetupOnError (see page 334)	TIBSetupOnError is the type of event handlers that respond to installation errors.
TIBSetupOnStatus (see page 334)	TIBSetupOnStatus is the type of event handlers that respond during the InterBase installation or de-installation process.
TIBSetupOnWarning (see page 335)	TIBSetupOnWarning is the type of event handlers that respond to installation warnings.
TIscError (see page 335)	This is type IBInstall.TIscError.
TMainOptions (see page 336)	This is type IBInstall.TMainOptions.

9.3.1 IBInstall.TCmdOptions Type

Pascal

```
TCmdOptions = set of TCmdOption;
```

C++

```
TCmdOption TCmdOptions;
```

File

IBInstall

Description

This is type IBInstall.TCmdOptions.

9.3.2 IBInstall.TConnectivityOptions Type

Pascal

```
TConnectivityOptions = set of TConnectivityOption;
```

C++

```
TConnectivityOption TConnectivityOptions;
```

File

IBInstall

Description

This is type IBInstall.TConnectivityOptions.

9.3.3 IBInstall.TExamplesOptions Type

Pascal

```
TExamplesOptions = set of TExamplesOption;
```

C++

```
TExamplesOption TExamplesOptions;
```

File

IBInstall

Description

This is type IBInstall.TExamplesOptions.

9.3.4 IBInstall.TIBSetupOnError Type

TIBSetupOnError is the type of event handlers that respond to installation errors.

Pascal

```
TIBSetupOnError = function (Sender : TObject; IscCode : TIscError; ErrorMessage, ErrorComment
: string): TErrorResult of object;
```

C++

```
(Sender : TObject; IscCode : TIscError; ErrorMessage, ErrorComment : string): TErrorResult (
TIBSetupOnError)();
```

File

IBInstall

Description

Sender is the TIBSetup (see page 317) component that encounters the exception.

IscCode is the error code for the problem that was encounted.

ErrorMessage is the error message fro the problem that occurred.

ErrorComment returns a string that will be displayed to the user.

The event handler is expected to return isc_fp_install_continue, isc_install_abort or isc_fp_install_retry.

9.3.5 IBInstall.TIBSetupOnStatus Type

TIBSetupOnStatus is the type of event handlers that respond during the InterBase installation or de-installation process.

Pascal

```
TIBSetupOnStatus = function (Sender : TObject; StatusComment : string): TStatusResult of
object;
```

C++

```
(Sender : TObject; StatusComment : string): TStatusResult ( TIBSetupOnStatus)();
```

File

IBInstall

Description

Sender is the TIBSetup (see page 317) component that is installing or uninstalling.

StatusComment is a short string that indicates what operation is currently occurring. This can be used to supply the user with feedback on the installation process.

The event handler can return isc_fp_install_continue if the installation should continue, or isc_install_abort if the user cancels the installation process.

9.3.6 IBInstall.TIBSetupOnWarning Type

TIBSetupOnWarning is the type of event handlers that respond to installation warnings.

Pascal

```
TIBSetupOnWarning = function (Sender :TObject; WarningCode: TIscError; WarningMessage :
string): TWarningResult of object;
```

C++

```
(Sender :TObject; WarningCode: TIscError; WarningMessage : string): TWarningResult (
TIBSetupOnWarning)();
```

File

IBInstall

Description

Sender is the TIBSetup (see page 317) component that encounters the warning.

WarningCode is the warning code for the situation that was encounted.

WarningMessage is the warning message for the situation that occurred.

The event handler must return either isc_fp_install_continue or isc_install_abort.

9.3.7 IBInstall.TIscError Type

Pascal

```
TIscError = MSG_NO;
```

C++

```
MSG_NO TIscError;
```

File

IBInstall

Description

This is type IBInstall.TIscError.

9.3.8 IBInstall.TMainOptions Type

Pascal

```
TMainOptions = set of TMainOption;
```

C++

```
TMainOption TMainOptions;
```

File

IBInstall

Description

This is type IBInstall.TMainOptions.

10 IBIntf Namespace

This is namespace IBIntf.

Topics

Name	Description
Functions (see page 340)	The following table lists functions in this documentation.
Variables (see page 351)	The following table lists variables in this documentation.

Functions

	Name	Description
	CheckIBInstallLoaded (see page 340)	Attempts to load IBInstall.DLL into memory if it is not already loaded.
	CheckIBLoaded (see page 341)	Attempts to load GDS32.DLL into memory if it is not already loaded.
	FreeIBInstallLibrary (see page 341)	Frees the IBInstall.DLL from memory.
	FreeIBLibrary (see page 341)	Frees the GDS32.DLL from memory.
	GetIBClientVersion (see page 342)	Returns the major version number of the InterBase client.
	isc_decode_sql_date_stub (see page 342)	This is function IBIntf.isc_decode_sql_date_stub.
	isc_decode_sql_time_stub (see page 342)	This is function IBIntf.isc_decode_sql_time_stub.
	isc_decode_timestamp_stub (see page 343)	This is function IBIntf.isc_decode_timestamp_stub.
	isc_encode_sql_date_stub (see page 343)	This is function IBIntf.isc_encode_sql_date_stub.
	isc_encode_sql_time_stub (see page 343)	This is function IBIntf.isc_encode_sql_time_stub.
	isc_encode_timestamp_stub (see page 344)	This is function IBIntf.isc_encode_timestamp_stub.
	isc_install_clear_options_stub (see page 344)	This is function IBIntf.isc_install_clear_options_stub.
	isc_install_execute_stub (see page 344)	This is function IBIntf.isc_install_execute_stub.
	isc_install_get_info_stub (see page 345)	This is function IBIntf.isc_install_get_info_stub.
	isc_install_get_message_stub (see page 345)	This is function IBIntf.isc_install_get_message_stub.
	isc_install_load_external_text_stub (see page 345)	This is function IBIntf.isc_install_load_external_text_stub.
	isc_install_precheck_stub (see page 346)	This is function IBIntf.isc_install_precheck_stub.
	isc_install_set_option_stub (see page 346)	This is function IBIntf.isc_install_set_option_stub.
	isc_install_unset_option_stub (see page 346)	This is function IBIntf.isc_install_unset_option_stub.
	isc_rollback_retaining_stub (see page 347)	This is function IBIntf.isc_rollback_retaining_stub.
	isc_service_attach_stub (see page 347)	This is function IBIntf.isc_service_attach_stub.
	isc_service_detach_stub (see page 347)	This is function IBIntf.isc_service_detach_stub.
	isc_service_query_stub (see page 348)	This is function IBIntf.isc_service_query_stub.
	isc_service_start_stub (see page 348)	This is function IBIntf.isc_service_start_stub.
	isc_uninstall_execute_stub (see page 348)	This is function IBIntf.isc_uninstall_execute_stub.
	isc_uninstall_precheck_stub (see page 349)	This is function IBIntf.isc_uninstall_precheck_stub.

	LoadIBInstallLibrary (see page 349)	Loads IBInstall.DLL into memory.
	LoadIBLibrary (see page 349)	Loads GDS32.DLL into memory.
	TryIBLoad (see page 350)	This is function IBIntf.TryIBLoad.

Variables

Name	Description
BLOB_get (see page 352)	This is variable IBIntf.BLOB_get.
BLOB_put (see page 352)	This is variable IBIntf.BLOB_put.
isc_add_user (see page 353)	This is variable IBIntf.isc_add_user.
isc_attach_database (see page 353)	This is variable IBIntf.isc_attach_database.
isc_blob_info (see page 353)	This is variable IBIntf.isc_blob_info.
isc_cancel_events (see page 354)	This is variable IBIntf.isc_cancel_events.
isc_close_blob (see page 354)	This is variable IBIntf.isc_close_blob.
isc_commit_retaining (see page 354)	This is variable IBIntf.isc_commit_retaining.
isc_commit_transaction (see page 355)	This is variable IBIntf.isc_commit_transaction.
isc_create_blob2 (see page 355)	This is variable IBIntf.isc_create_blob2.
isc_database_info (see page 355)	This is variable IBIntf.isc_database_info.
isc_decode_date (see page 356)	This is variable IBIntf.isc_decode_date.
isc_decode_sql_date (see page 356)	This is variable IBIntf.isc_decode_sql_date.
isc_decode_sql_time (see page 356)	This is variable IBIntf.isc_decode_sql_time.
isc_decode_timestamp (see page 357)	This is variable IBIntf.isc_decode_timestamp.
isc_delete_user (see page 357)	This is variable IBIntf.isc_delete_user.
isc_detach_database (see page 357)	This is variable IBIntf.isc_detach_database.
isc_drop_database (see page 358)	This is variable IBIntf.isc_drop_database.
isc_dsql_alloc_statement2 (see page 358)	This is variable IBIntf.isc_dsql_alloc_statement2.
isc_dsql_describe (see page 358)	This is variable IBIntf.isc_dsql_describe.
isc_dsql_describe_bind (see page 359)	This is variable IBIntf.isc_dsql_describe_bind.
isc_dsql_execute (see page 359)	This is variable IBIntf.isc_dsql_execute.
isc_dsql_execute_immediate (see page 359)	This is variable IBIntf.isc_dsql_execute_immediate.
isc_dsql_execute2 (see page 360)	This is variable IBIntf.isc_dsql_execute2.
isc_dsql_fetch (see page 360)	This is variable IBIntf.isc_dsql_fetch.
isc_dsql_free_statement (see page 360)	This is variable IBIntf.isc_dsql_free_statement.
isc_dsql_prepare (see page 361)	This is variable IBIntf.isc_dsql_prepare.
isc_dsql_set_cursor_name (see page 361)	This is variable IBIntf.isc_dsql_set_cursor_name.
isc_dsql_sql_info (see page 361)	This is variable IBIntf.isc_dsql_sql_info.
isc_encode_date (see page 362)	This is variable IBIntf.isc_encode_date.
isc_encode_sql_date (see page 362)	This is variable IBIntf.isc_encode_sql_date.
isc_encode_sql_time (see page 362)	This is variable IBIntf.isc_encode_sql_time.
isc_encode_timestamp (see page 363)	This is variable IBIntf.isc_encode_timestamp.
isc_event_block (see page 363)	This is variable IBIntf.isc_event_block.
isc_event_counts (see page 363)	This is variable IBIntf.isc_event_counts.
isc_free (see page 364)	This is variable IBIntf.isc_free.
isc_get_segment (see page 364)	This is variable IBIntf.isc_get_segment.
isc_install_clear_options (see page 364)	This is variable IBIntf.isc_install_clear_options.
isc_install_execute (see page 365)	This is variable IBIntf.isc_install_execute.
isc_install_get_info (see page 365)	This is variable IBIntf.isc_install_get_info.
isc_install_get_message (see page 365)	This is variable IBIntf.isc_install_get_message.
isc_install_load_external_text (see page 366)	This is variable IBIntf.isc_install_load_external_text.
isc_install_precheck (see page 366)	This is variable IBIntf.isc_install_precheck.
isc_install_set_option (see page 366)	This is variable IBIntf.isc_install_set_option.
isc_install_unset_option (see page 367)	This is variable IBIntf.isc_install_unset_option.
isc_interprete (see page 367)	This is variable IBIntf.isc_interprete.
isc_modify_user (see page 367)	This is variable IBIntf.isc_modify_user.
isc_open_blob2 (see page 368)	This is variable IBIntf.isc_open_blob2.

isc_prepare_transaction (see page 368)	This is variable IBIntf.isc_prepare_transaction.
isc_prepare_transaction2 (see page 368)	This is variable IBIntf.isc_prepare_transaction2.
isc_put_segment (see page 369)	This is variable IBIntf.isc_put_segment.
isc_que_events (see page 369)	This is variable IBIntf.isc_que_events.
isc_rollback_retaining (see page 369)	This is variable IBIntf.isc_rollback_retaining.
isc_rollback_transaction (see page 370)	This is variable IBIntf.isc_rollback_transaction.
isc_service_attach (see page 370)	This is variable IBIntf.isc_service_attach.
isc_service_detach (see page 370)	This is variable IBIntf.isc_service_detach.
isc_service_query (see page 371)	This is variable IBIntf.isc_service_query.
isc_service_start (see page 371)	This is variable IBIntf.isc_service_start.
isc_sql_interprete (see page 371)	This is variable IBIntf.isc_sql_interprete.
isc_sqlcode (see page 372)	This is variable IBIntf.isc_sqlcode.
isc_start_multiple (see page 372)	This is variable IBIntf.isc_start_multiple.
isc_uninstall_execute (see page 372)	This is variable IBIntf.isc_uninstall_execute.
isc_uninstall_precheck (see page 373)	This is variable IBIntf.isc_uninstall_precheck.
isc_vax_integer (see page 373)	This is variable IBIntf.isc_vax_integer.

10.1 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	CheckIBInstallLoaded (see page 340)	Attempts to load IBInstall.DLL into memory if it is not already loaded.
	CheckIBLoaded (see page 341)	Attempts to load GDS32.DLL into memory if it is not already loaded.
	FreeIBInstallLibrary (see page 341)	Frees the IBInstall.DLL from memory.
	FreeIBLibrary (see page 341)	Frees the GDS32.DLL from memory.
	GetIBClientVersion (see page 342)	Returns the major version number of the InterBase client.
	isc_decode_sql_date_stub (see page 342)	This is function IBIntf.isc_decode_sql_date_stub.
	isc_decode_sql_time_stub (see page 342)	This is function IBIntf.isc_decode_sql_time_stub.
	isc_decode_timestamp_stub (see page 343)	This is function IBIntf.isc_decode_timestamp_stub.
	isc_encode_sql_date_stub (see page 343)	This is function IBIntf.isc_encode_sql_date_stub.
	isc_encode_sql_time_stub (see page 343)	This is function IBIntf.isc_encode_sql_time_stub.
	isc_encode_timestamp_stub (see page 344)	This is function IBIntf.isc_encode_timestamp_stub.
	isc_install_clear_options_stub (see page 344)	This is function IBIntf.isc_install_clear_options_stub.
	isc_install_execute_stub (see page 344)	This is function IBIntf.isc_install_execute_stub.
	isc_install_get_info_stub (see page 345)	This is function IBIntf.isc_install_get_info_stub.
	isc_install_get_message_stub (see page 345)	This is function IBIntf.isc_install_get_message_stub.
	isc_install_load_external_text_stub (see page 345)	This is function IBIntf.isc_install_load_external_text_stub.
	isc_install_precheck_stub (see page 346)	This is function IBIntf.isc_install_precheck_stub.
	isc_install_set_option_stub (see page 346)	This is function IBIntf.isc_install_set_option_stub.
	isc_install_unset_option_stub (see page 346)	This is function IBIntf.isc_install_unset_option_stub.
	isc_rollback_retaining_stub (see page 347)	This is function IBIntf.isc_rollback_retaining_stub.
	isc_service_attach_stub (see page 347)	This is function IBIntf.isc_service_attach_stub.
	isc_service_detach_stub (see page 347)	This is function IBIntf.isc_service_detach_stub.
	isc_service_query_stub (see page 348)	This is function IBIntf.isc_service_query_stub.
	isc_service_start_stub (see page 348)	This is function IBIntf.isc_service_start_stub.
	isc_uninstall_execute_stub (see page 348)	This is function IBIntf.isc_uninstall_execute_stub.
	isc_uninstall_precheck_stub (see page 349)	This is function IBIntf.isc_uninstall_precheck_stub.
	LoadIBInstallLibrary (see page 349)	Loads IBInstall.DLL into memory.
	LoadIBLibrary (see page 349)	Loads GDS32.DLL into memory.
	TryIBLoad (see page 350)	This is function IBIntf.TryIBLoad.

10.1.1 IBIntf.CheckIBInstallLoaded Function

Attempts to load IBInstall.DLL into memory if it is not already loaded.

Pascal

```
procedure CheckIBInstallLoaded;
```

C++

```
CheckIBInstallLoaded();
```

File

IBIntf

Description

Use the CheckIBInstallLibrary procedure to load IBInstall.DLL into memory if it is not already loaded. If unsuccessful, this procedure raises an exception.

10.1.2 IBIntf.CheckIBLoaded Function

Attempts to load GDS32.DLL into memory if it is not already loaded.

Pascal

```
procedure CheckIBLoaded;
```

C++

```
CheckIBLoaded();
```

File

IBIntf

Description

Use the CheckIBInstallLibrary procedure to load GDS32.DLL into memory if it is not already loaded. If unsuccessful, this procedure raises an exception.

10.1.3 IBIntf.FreeIBInstallLibrary Function

Frees the IBInstall.DLL from memory.

Pascal

```
procedure FreeIBInstallLibrary;
```

C++

```
FreeIBInstallLibrary();
```

File

IBIntf

Description

Use the FreeIBInstallLibrary procedure to free the previously loaded the IBInstall.DLL from memory.

10.1.4 IBIntf.FreeIBLibrary Function

Frees the GDS32.DLL from memory.

Pascal

```
procedure FreeIBLibrary;
```

C++

```
FreeIBLibrary();
```

File

IBIntf

Description

Use the FreeIBInstallLibrary (see page 341) procedure to free the previously loaded the GDS32.DLL from memory.

10.1.5 IBIntf.GetIBClientVersion Function

Returns the major version number of the InterBase client.

Pascal

```
function GetIBClientVersion: Integer;
```

C++

```
int GetIBClientVersion();
```

File

IBIntf

Description

Use the GetIBClientVersion function to retrieve the major version number of the InterBase client.

10.1.6 IBIntf.isc_decode_sql_date_stub Function

Pascal

```
procedure isc_decode_sql_date_stub(ib_date: PISC_DATE; tm_date: PCTimeStructure); stdcall;
```

C++

```
__stdcall isc_decode_sql_date_stub(PISC_DATE ib_date, PCTimeStructure tm_date);
```

File

IBIntf

Description

This is function IBIntf.isc_decode_sql_date_stub.

10.1.7 IBIntf.isc_decode_sql_time_stub Function

Pascal

```
procedure isc_decode_sql_time_stub(ib_time: PISC_TIME; tm_date: PCTimeStructure); stdcall;
```

C++

```
__stdcall isc_decode_sql_time_stub(PISC_TIME ib_time, PCTimeStructure tm_date);
```

File

IBIntf

Description

This is function IBIntf.isc_decode_sql_time_stub.

10.1.8 IBIntf.isc_decode_timestamp_stub Function

Pascal

```
procedure isc_decode_timestamp_stub(ib_timestamp: PISC_TIMESTAMP; tm_date: PCTimeStructure);
stdcall;
```

C++

```
__stdcall isc_decode_timestamp_stub(PISC_TIMESTAMP ib_timestamp, PCTimeStructure tm_date);
```

File

IBIntf

Description

This is function IBIntf.isc_decode_timestamp_stub.

10.1.9 IBIntf.isc_encode_sql_date_stub Function

Pascal

```
procedure isc_encode_sql_date_stub(tm_date: PCTimeStructure; ib_date: PISC_DATE); stdcall;
```

C++

```
__stdcall isc_encode_sql_date_stub(PCTimeStructure tm_date, PISC_DATE ib_date);
```

File

IBIntf

Description

This is function IBIntf.isc_encode_sql_date_stub.

10.1.10 IBIntf.isc_encode_sql_time_stub Function

Pascal

```
procedure isc_encode_sql_time_stub(tm_date: PCTimeStructure; ib_time: PISC_TIME); stdcall;
```

C++

```
__stdcall isc_encode_sql_time_stub(PCTimeStructure tm_date, PISC_TIME ib_time);
```

File

IBIntf

Description

This is function IBIntf.isc_encode_sql_time_stub.

10.1.11 IBIntf.isc_encode_timestamp_stub Function

Pascal

```
procedure isc_encode_timestamp_stub(tm_date: PCTimeStructure; ib_timestamp: PISC_TIMESTAMP);
stdcall;
```

C++

```
__stdcall isc_encode_timestamp_stub(PCTimeStructure tm_date, PISC_TIMESTAMP ib_timestamp);
```

File

IBIntf

Description

This is function IBIntf.isc_encode_timestamp_stub.

10.1.12 IBIntf.isc_install_clear_options_stub Function

Pascal

```
function isc_install_clear_options_stub(hOption: POPTIONS_HANDLE): MSG_NO; stdcall;
```

C++

```
__stdcall MSG_NO isc_install_clear_options_stub(POPTIONS_HANDLE hOption);
```

File

IBIntf

Description

This is function IBIntf.isc_install_clear_options_stub.

10.1.13 IBIntf.isc_install_execute_stub Function

Pascal

```
function isc_install_execute_stub(hOption: OPTIONS_HANDLE; src_dir: TEXT; dest_dir: TEXT;
status_func: FP_STATUS; status_data: pointer; error_func: FP_ERROR; error_data: pointer;
uninstal_file_name: TEXT): MSG_NO; stdcall;
```

C++

```
__stdcall MSG_NO isc_install_execute_stub(OPTIONS_HANDLE hOption, TEXT src_dir, TEXT dest_dir,
FP_STATUS status_func, void * status_data, FP_ERROR error_func, void * error_data, TEXT
uninstal_file_name);
```

File

IBIntf

Description

This is function IBIntf.isc_install_execute_stub.

10.1.14 IBIntf.isc_install_get_info_stub Function

Pascal

```
function isc_install_get_info_stub(info_type: integer; option: OPT; info_buffer: Pointer;
buf_len: Cardinal): MSG_NO; stdcall;
```

C++

```
__stdcall MSG_NO isc_install_get_info_stub(int info_type, OPT option, void * info_buffer,
unsigned buf_len);
```

File

IBIntf

Description

This is function IBIntf.isc_install_get_info_stub.

10.1.15 IBIntf.isc_install_get_message_stub Function

Pascal

```
function isc_install_get_message_stub(hOption: OPTIONS_HANDLE; message_no: MSG_NO;
message_txt: Pointer; message_len: Cardinal): MSG_NO; stdcall;
```

C++

```
__stdcall MSG_NO isc_install_get_message_stub(OPTIONS_HANDLE hOption, MSG_NO message_no, void
* message_txt, unsigned message_len);
```

File

IBIntf

Description

This is function IBIntf.isc_install_get_message_stub.

10.1.16 IBIntf.isc_install_load_external_text_stub Function

Pascal

```
function isc_install_load_external_text_stub(msg_file_name: TEXT): MSG_NO; stdcall;
```

C++

```
__stdcall MSG_NO isc_install_load_external_text_stub(TEXT msg_file_name);
```

File

IBIntf

Description

This is function IBIntf.isc_install_load_external_text_stub.

10.1.17 IBIntf.isc_install_precheck_stub Function

Pascal

```
function isc_install_precheck_stub(hOption: OPTIONS_HANDLE; src_dir: TEXT; dest_dir: TEXT):
MSG_NO; stdcall;
```

C++

```
__stdcall MSG_NO isc_install_precheck_stub(OPTIONS_HANDLE hOption, TEXT src_dir, TEXT
dest_dir);
```

File

IBIntf

Description

This is function IBIntf.isc_install_precheck_stub.

10.1.18 IBIntf.isc_install_set_option_stub Function

Pascal

```
function isc_install_set_option_stub(hOption: POPTIONS_HANDLE; option: OPT): MSG_NO; stdcall;
```

C++

```
__stdcall MSG_NO isc_install_set_option_stub(POPTIONS_HANDLE hOption, OPT option);
```

File

IBIntf

Description

This is function IBIntf.isc_install_set_option_stub.

10.1.19 IBIntf.isc_install_unset_option_stub Function

Pascal

```
function isc_install_unset_option_stub(hOption: POPTIONS_HANDLE; option: OPT): MSG_NO; stdcall;
```

C++

```
__stdcall MSG_NO isc_install_unset_option_stub(POPTIONS_HANDLE hOption, OPT option);
```

File

IBIntf

Description

This is function IBIntf.isc_install_unset_option_stub.

10.1.20 IBIntf.isc_rollback_retaining_stub Function

Pascal

```
function isc_rollback_retaining_stub(status_vector: PISC_STATUS; tran_handle: PISC_TR_HANDLE):
ISC_STATUS; stdcall;
```

C++

```
__stdcall ISC_STATUS isc_rollback_retaining_stub(PISC_STATUS status_vector, PISC_TR_HANDLE
tran_handle);
```

File

IBIntf

Description

This is function IBIntf.isc_rollback_retaining_stub.

10.1.21 IBIntf.isc_service_attach_stub Function

Pascal

```
function isc_service_attach_stub(status_vector: PISC_STATUS; isc_arg2: UShort; isc_arg3:
PChar; service_handle: PISC_SVC_HANDLE; isc_arg5: UShort; isc_arg6: PChar): ISC_STATUS;
stdcall;
```

C++

```
__stdcall ISC_STATUS isc_service_attach_stub(PISC_STATUS status_vector, UShort isc_arg2, const
char * isc_arg3, PISC_SVC_HANDLE service_handle, UShort isc_arg5, const char * isc_arg6);
```

File

IBIntf

Description

This is function IBIntf.isc_service_attach_stub.

10.1.22 IBIntf.isc_service_detach_stub Function

Pascal

```
function isc_service_detach_stub(status_vector: PISC_STATUS; service_handle: PISC_SVC_HANDLE):
ISC_STATUS; stdcall;
```

C++

```
__stdcall ISC_STATUS isc_service_detach_stub(PISC_STATUS status_vector, PISC_SVC_HANDLE
service_handle);
```

File

IBIntf

Description

This is function IBIntf.isc_service_detach_stub.

10.1.23 IBIntf.isc_service_query_stub Function

Pascal

```
function isc_service_query_stub(status_vector: PISC_STATUS; service_handle: PISC_SVC_HANDLE;
recv_handle: PISC_SVC_HANDLE; isc_arg4: UShort; isc_arg5: PChar; isc_arg6: UShort; isc_arg7:
PChar; isc_arg8: UShort; isc_arg9: PChar): ISC_STATUS; stdcall;
```

C++

```
__stdcall ISC_STATUS isc_service_query_stub(PISC_STATUS status_vector, PISC_SVC_HANDLE
service_handle, PISC_SVC_HANDLE recv_handle, UShort isc_arg4, const char * isc_arg5, UShort
isc_arg6, const char * isc_arg7, UShort isc_arg8, const char * isc_arg9);
```

File

IBIntf

Description

This is function IBIntf.isc_service_query_stub.

10.1.24 IBIntf.isc_service_start_stub Function

Pascal

```
function isc_service_start_stub(status_vector: PISC_STATUS; service_handle: PISC_SVC_HANDLE;
recv_handle: PISC_SVC_HANDLE; isc_arg4: UShort; isc_arg5: PChar): ISC_STATUS; stdcall;
```

C++

```
__stdcall ISC_STATUS isc_service_start_stub(PISC_STATUS status_vector, PISC_SVC_HANDLE
service_handle, PISC_SVC_HANDLE recv_handle, UShort isc_arg4, const char * isc_arg5);
```

File

IBIntf

Description

This is function IBIntf.isc_service_start_stub.

10.1.25 IBIntf.isc_uninstall_execute_stub Function

Pascal

```
function isc_uninstall_execute_stub(uninstall_file_name: TEXT; status_func: FP_STATUS;
status_data: pointer; error_func: FP_ERROR; error_data: pointer): MSG_NO; stdcall;
```

C++

```
__stdcall MSG_NO isc_uninstall_execute_stub(TEXT uninstall_file_name, FP_STATUS status_func,
void * status_data, FP_ERROR error_func, void * error_data);
```

File

IBIntf

Description

This is function IBIntf.isc_uninstall_execute_stub.

10.1.26 IBIntf.isc_uninstall_precheck_stub Function

Pascal

```
function isc_uninstall_precheck_stub(uninstall_file_name: TEXT): MSG_NO; stdcall;
```

C++

```
__stdcall MSG_NO isc_uninstall_precheck_stub(TEXT uninstall_file_name);
```

File

IBIntf

Description

This is function IBIntf.isc_uninstall_precheck_stub.

10.1.27 IBIntf.LoadIBInstallLibrary Function

Loads IBInstall.DLL into memory.

Pascal

```
procedure LoadIBInstallLibrary;
```

C++

```
LoadIBInstallLibrary();
```

File

IBIntf

Description

Use the LoadIBInstallLibrary procedure to load the IBInstall.DLL into memory and resolve the respective entry points into to the respective procedure pointer in the unit.

10.1.28 IBIntf.LoadIBLibrary Function

Loads GDS32.DLL into memory.

Pascal

```
procedure LoadIBLibrary;
```

C++

```
LoadIBLibrary();
```

File

IBIntf

Description

Use the LoadIBLibrary procedure to load the GDS32.DLL into memory and resolve the respective entry points into to the respective procedure pointer in the unit. In addition, it loads the InterBase entry points, if they are available.

10.1.29 IBIntf.TryIBLoad Function

Pascal

```
function TryIBLoad: Boolean;
```

C++

```
Boolean TryIBLoad();
```

File

IBIntf

Description

This is function IBIntf.TryIBLoad.

10.2 Variables

The following table lists variables in this documentation.

Variables

Name	Description
BLOB_get (see page 352)	This is variable IBIntf.BLOB_get.
BLOB_put (see page 352)	This is variable IBIntf.BLOB_put.
isc_add_user (see page 353)	This is variable IBIntf.isc_add_user.
isc_attach_database (see page 353)	This is variable IBIntf.isc_attach_database.
isc_blob_info (see page 353)	This is variable IBIntf.isc_blob_info.
isc_cancel_events (see page 354)	This is variable IBIntf.isc_cancel_events.
isc_close_blob (see page 354)	This is variable IBIntf.isc_close_blob.
isc_commit_retaining (see page 354)	This is variable IBIntf.isc_commit_retaining.
isc_commit_transaction (see page 355)	This is variable IBIntf.isc_commit_transaction.
isc_create_blob2 (see page 355)	This is variable IBIntf.isc_create_blob2.
isc_database_info (see page 355)	This is variable IBIntf.isc_database_info.
isc_decode_date (see page 356)	This is variable IBIntf.isc_decode_date.
isc_decode_sql_date (see page 356)	This is variable IBIntf.isc_decode_sql_date.
isc_decode_sql_time (see page 356)	This is variable IBIntf.isc_decode_sql_time.
isc_decode_timestamp (see page 357)	This is variable IBIntf.isc_decode_timestamp.
isc_delete_user (see page 357)	This is variable IBIntf.isc_delete_user.
isc_detach_database (see page 357)	This is variable IBIntf.isc_detach_database.
isc_drop_database (see page 358)	This is variable IBIntf.isc_drop_database.
isc_dsql_alloc_statement2 (see page 358)	This is variable IBIntf.isc_dsql_alloc_statement2.
isc_dsql_describe (see page 358)	This is variable IBIntf.isc_dsql_describe.
isc_dsql_describe_bind (see page 359)	This is variable IBIntf.isc_dsql_describe_bind.
isc_dsql_execute (see page 359)	This is variable IBIntf.isc_dsql_execute.
isc_dsql_execute_immediate (see page 359)	This is variable IBIntf.isc_dsql_execute_immediate.
isc_dsql_execute2 (see page 360)	This is variable IBIntf.isc_dsql_execute2.
isc_dsql_fetch (see page 360)	This is variable IBIntf.isc_dsql_fetch.
isc_dsql_free_statement (see page 360)	This is variable IBIntf.isc_dsql_free_statement.
isc_dsql_prepare (see page 361)	This is variable IBIntf.isc_dsql_prepare.
isc_dsql_set_cursor_name (see page 361)	This is variable IBIntf.isc_dsql_set_cursor_name.
isc_dsql_sql_info (see page 361)	This is variable IBIntf.isc_dsql_sql_info.
isc_encode_date (see page 362)	This is variable IBIntf.isc_encode_date.
isc_encode_sql_date (see page 362)	This is variable IBIntf.isc_encode_sql_date.
isc_encode_sql_time (see page 362)	This is variable IBIntf.isc_encode_sql_time.
isc_encode_timestamp (see page 363)	This is variable IBIntf.isc_encode_timestamp.
isc_event_block (see page 363)	This is variable IBIntf.isc_event_block.
isc_event_counts (see page 363)	This is variable IBIntf.isc_event_counts.
isc_free (see page 364)	This is variable IBIntf.isc_free.
isc_get_segment (see page 364)	This is variable IBIntf.isc_get_segment.
isc_install_clear_options (see page 364)	This is variable IBIntf.isc_install_clear_options.
isc_install_execute (see page 365)	This is variable IBIntf.isc_install_execute.
isc_install_get_info (see page 365)	This is variable IBIntf.isc_install_get_info.
isc_install_get_message (see page 365)	This is variable IBIntf.isc_install_get_message.
isc_install_load_external_text (see page 366)	This is variable IBIntf.isc_install_load_external_text.
isc_install_precheck (see page 366)	This is variable IBIntf.isc_install_precheck.
isc_install_set_option (see page 366)	This is variable IBIntf.isc_install_set_option.

isc_install_unset_option (see page 367)	This is variable IBIntf.isc_install_unset_option.
isc_interprete (see page 367)	This is variable IBIntf.isc_interprete.
isc_modify_user (see page 367)	This is variable IBIntf.isc_modify_user.
isc_open_blob2 (see page 368)	This is variable IBIntf.isc_open_blob2.
isc_prepare_transaction (see page 368)	This is variable IBIntf.isc_prepare_transaction.
isc_prepare_transaction2 (see page 368)	This is variable IBIntf.isc_prepare_transaction2.
isc_put_segment (see page 369)	This is variable IBIntf.isc_put_segment.
isc_que_events (see page 369)	This is variable IBIntf.isc_que_events.
isc_rollback_retaining (see page 369)	This is variable IBIntf.isc_rollback_retaining.
isc_rollback_transaction (see page 370)	This is variable IBIntf.isc_rollback_transaction.
isc_service_attach (see page 370)	This is variable IBIntf.isc_service_attach.
isc_service_detach (see page 370)	This is variable IBIntf.isc_service_detach.
isc_service_query (see page 371)	This is variable IBIntf.isc_service_query.
isc_service_start (see page 371)	This is variable IBIntf.isc_service_start.
isc_sql_interprete (see page 371)	This is variable IBIntf.isc_sql_interprete.
isc_sqlcode (see page 372)	This is variable IBIntf.isc_sqlcode.
isc_start_multiple (see page 372)	This is variable IBIntf.isc_start_multiple.
isc_uninstall_execute (see page 372)	This is variable IBIntf.isc_uninstall_execute.
isc_uninstall_precheck (see page 373)	This is variable IBIntf.isc_uninstall_precheck.
isc_vax_integer (see page 373)	This is variable IBIntf.isc_vax_integer.

10.2.1 IBIntf.BLOB_get Variable

Pascal

```
BLOB_get: TBLOB_get;
```

C++

```
TBLOB_get BLOB_get;
```

File

IBIntf

Description

This is variable IBIntf.BLOB_get.

10.2.2 IBIntf.BLOB_put Variable

Pascal

```
BLOB_put: TBLOB_put;
```

C++

```
TBLOB_put BLOB_put;
```

File

IBIntf

Description

This is variable IBIntf.BLOB_put.

10.2.3 IBIntf.isc_add_user Variable

Pascal

```
isc_add_user: Tisc_add_user;
```

C++

```
Tisc_add_user isc_add_user;
```

File

IBIntf

Description

This is variable IBIntf.isc_add_user.

10.2.4 IBIntf.isc_attach_database Variable

Pascal

```
isc_attach_database: Tisc_attach_database;
```

C++

```
Tisc_attach_database isc_attach_database;
```

File

IBIntf

Description

This is variable IBIntf.isc_attach_database.

10.2.5 IBIntf.isc_blob_info Variable

Pascal

```
isc_blob_info: Tisc_blob_info;
```

C++

```
Tisc_blob_info isc_blob_info;
```

File

IBIntf

Description

This is variable IBIntf.isc_blob_info.

10.2.6 IBIntf.isc_cancel_events Variable

Pascal

```
isc_cancel_events: Tisc_cancel_events;
```

C++

```
Tisc_cancel_events isc_cancel_events;
```

File

IBIntf

Description

This is variable IBIntf.isc_cancel_events.

10.2.7 IBIntf.isc_close_blob Variable

Pascal

```
isc_close_blob: Tisc_close_blob;
```

C++

```
Tisc_close_blob isc_close_blob;
```

File

IBIntf

Description

This is variable IBIntf.isc_close_blob.

10.2.8 IBIntf.isc_commit_retaining Variable

Pascal

```
isc_commit_retaining: Tisc_commit_retaining;
```

C++

```
Tisc_commit_retaining isc_commit_retaining;
```

File

IBIntf

Description

This is variable IBIntf.isc_commit_retaining.

10.2.9 IBIntf.isc_commit_transaction Variable

Pascal

```
isc_commit_transaction: Tisc_commit_transaction;
```

C++

```
Tisc_commit_transaction isc_commit_transaction;
```

File

IBIntf

Description

This is variable IBIntf.isc_commit_transaction.

10.2.10 IBIntf.isc_create_blob2 Variable

Pascal

```
isc_create_blob2: Tisc_create_blob2;
```

C++

```
Tisc_create_blob2 isc_create_blob2;
```

File

IBIntf

Description

This is variable IBIntf.isc_create_blob2.

10.2.11 IBIntf.isc_database_info Variable

Pascal

```
isc_database_info: Tisc_database_info;
```

C++

```
Tisc_database_info isc_database_info;
```

File

IBIntf

Description

This is variable IBIntf.isc_database_info.

10.2.12 IBIntf.isc_decode_date Variable

Pascal

```
isc_decode_date: Tisc_decode_date;
```

C++

```
Tisc_decode_date isc_decode_date;
```

File

IBIntf

Description

This is variable IBIntf.isc_decode_date.

10.2.13 IBIntf.isc_decode_sql_date Variable

Pascal

```
isc_decode_sql_date: Tisc_decode_sql_date;
```

C++

```
Tisc_decode_sql_date isc_decode_sql_date;
```

File

IBIntf

Description

This is variable IBIntf.isc_decode_sql_date.

10.2.14 IBIntf.isc_decode_sql_time Variable

Pascal

```
isc_decode_sql_time: Tisc_decode_sql_time;
```

C++

```
Tisc_decode_sql_time isc_decode_sql_time;
```

File

IBIntf

Description

This is variable IBIntf.isc_decode_sql_time.

10.2.15 IBIntf.isc_decode_timestamp Variable

Pascal

```
isc_decode_timestamp: Tisc_decode_timestamp;
```

C++

```
Tisc_decode_timestamp isc_decode_timestamp;
```

File

IBIntf

Description

This is variable IBIntf.isc_decode_timestamp.

10.2.16 IBIntf.isc_delete_user Variable

Pascal

```
isc_delete_user: Tisc_delete_user;
```

C++

```
Tisc_delete_user isc_delete_user;
```

File

IBIntf

Description

This is variable IBIntf.isc_delete_user.

10.2.17 IBIntf.isc_detach_database Variable

Pascal

```
isc_detach_database: Tisc_detach_database;
```

C++

```
Tisc_detach_database isc_detach_database;
```

File

IBIntf

Description

This is variable IBIntf.isc_detach_database.

10.2.18 IBIntf.isc_drop_database Variable

Pascal

```
isc_drop_database: Tisc_drop_database;
```

C++

```
Tisc_drop_database isc_drop_database;
```

File

IBIntf

Description

This is variable IBIntf.isc_drop_database.

10.2.19 IBIntf.isc_dsql_alloc_statement2 Variable

Pascal

```
isc_dsql_alloc_statement2: Tisc_dsql_alloc_statement2;
```

C++

```
Tisc_dsql_alloc_statement2 isc_dsql_alloc_statement2;
```

File

IBIntf

Description

This is variable IBIntf.isc_dsql_alloc_statement2.

10.2.20 IBIntf.isc_dsql_describe Variable

Pascal

```
isc_dsql_describe: Tisc_dsql_describe;
```

C++

```
Tisc_dsql_describe isc_dsql_describe;
```

File

IBIntf

Description

This is variable IBIntf.isc_dsql_describe.

10.2.21 IBIntf.isc_dsql_describe_bind Variable

Pascal

```
isc_dsql_describe_bind: Tisc_dsql_describe_bind;
```

C++

```
Tisc_dsql_describe_bind isc_dsql_describe_bind;
```

File

IBIntf

Description

This is variable IBIntf.isc_dsql_describe_bind.

10.2.22 IBIntf.isc_dsql_execute Variable

Pascal

```
isc_dsql_execute: Tisc_dsql_execute;
```

C++

```
Tisc_dsql_execute isc_dsql_execute;
```

File

IBIntf

Description

This is variable IBIntf.isc_dsql_execute.

10.2.23 IBIntf.isc_dsql_execute_immediate Variable

Pascal

```
isc_dsql_execute_immediate: Tisc_dsql_execute_immediate;
```

C++

```
Tisc_dsql_execute_immediate isc_dsql_execute_immediate;
```

File

IBIntf

Description

This is variable IBIntf.isc_dsql_execute_immediate.

10.2.24 IBIntf.isc_dsql_execute2 Variable

Pascal

```
isc_dsql_execute2: Tisc_dsql_execute2;
```

C++

```
Tisc_dsql_execute2 isc_dsql_execute2;
```

File

IBIntf

Description

This is variable IBIntf.isc_dsql_execute2.

10.2.25 IBIntf.isc_dsql_fetch Variable

Pascal

```
isc_dsql_fetch: Tisc_dsql_fetch;
```

C++

```
Tisc_dsql_fetch isc_dsql_fetch;
```

File

IBIntf

Description

This is variable IBIntf.isc_dsql_fetch.

10.2.26 IBIntf.isc_dsql_free_statement Variable

Pascal

```
isc_dsql_free_statement: Tisc_dsql_free_statement;
```

C++

```
Tisc_dsql_free_statement isc_dsql_free_statement;
```

File

IBIntf

Description

This is variable IBIntf.isc_dsql_free_statement.

10.2.27 IBIntf.isc_dsql_prepare Variable

Pascal

```
isc_dsql_prepare: Tisc_dsql_prepare;
```

C++

```
Tisc_dsql_prepare isc_dsql_prepare;
```

File

IBIntf

Description

This is variable IBIntf.isc_dsql_prepare.

10.2.28 IBIntf.isc_dsql_set_cursor_name Variable

Pascal

```
isc_dsql_set_cursor_name: Tisc_dsql_set_cursor_name;
```

C++

```
Tisc_dsql_set_cursor_name isc_dsql_set_cursor_name;
```

File

IBIntf

Description

This is variable IBIntf.isc_dsql_set_cursor_name.

10.2.29 IBIntf.isc_dsql_sql_info Variable

Pascal

```
isc_dsql_sql_info: Tisc_dsql_sql_info;
```

C++

```
Tisc_dsql_sql_info isc_dsql_sql_info;
```

File

IBIntf

Description

This is variable IBIntf.isc_dsql_sql_info.

10.2.30 IBIntf.isc_encode_date Variable

Pascal

```
isc_encode_date: Tisc_encode_date;
```

C++

```
Tisc_encode_date isc_encode_date;
```

File

IBIntf

Description

This is variable IBIntf.isc_encode_date.

10.2.31 IBIntf.isc_encode_sql_date Variable

Pascal

```
isc_encode_sql_date: Tisc_encode_sql_date;
```

C++

```
Tisc_encode_sql_date isc_encode_sql_date;
```

File

IBIntf

Description

This is variable IBIntf.isc_encode_sql_date.

10.2.32 IBIntf.isc_encode_sql_time Variable

Pascal

```
isc_encode_sql_time: Tisc_encode_sql_time;
```

C++

```
Tisc_encode_sql_time isc_encode_sql_time;
```

File

IBIntf

Description

This is variable IBIntf.isc_encode_sql_time.

10.2.33 IBIntf.isc_encode_timestamp Variable

Pascal

```
isc_encode_timestamp: Tisc_encode_timestamp;
```

C++

```
Tisc_encode_timestamp isc_encode_timestamp;
```

File

IBIntf

Description

This is variable IBIntf.isc_encode_timestamp.

10.2.34 IBIntf.isc_event_block Variable

Pascal

```
isc_event_block: Tisc_event_block;
```

C++

```
Tisc_event_block isc_event_block;
```

File

IBIntf

Description

This is variable IBIntf.isc_event_block.

10.2.35 IBIntf.isc_event_counts Variable

Pascal

```
isc_event_counts: Tisc_event_counts;
```

C++

```
Tisc_event_counts isc_event_counts;
```

File

IBIntf

Description

This is variable IBIntf.isc_event_counts.

10.2.36 IBIntf.isc_free Variable

Pascal

```
isc_free: Tisc_free;
```

C++

```
Tisc_free isc_free;
```

File

IBIntf

Description

This is variable IBIntf.isc_free.

10.2.37 IBIntf.isc_get_segment Variable

Pascal

```
isc_get_segment: Tisc_get_segment;
```

C++

```
Tisc_get_segment isc_get_segment;
```

File

IBIntf

Description

This is variable IBIntf.isc_get_segment.

10.2.38 IBIntf.isc_install_clear_options Variable

Pascal

```
isc_install_clear_options: Tisc_install_clear_options;
```

C++

```
Tisc_install_clear_options isc_install_clear_options;
```

File

IBIntf

Description

This is variable IBIntf.isc_install_clear_options.

10.2.39 IBIntf.isc_install_execute Variable

Pascal

```
isc_install_execute: Tisc_install_execute;
```

C++

```
Tisc_install_execute isc_install_execute;
```

File

IBIntf

Description

This is variable IBIntf.isc_install_execute.

10.2.40 IBIntf.isc_install_get_info Variable

Pascal

```
isc_install_get_info: Tisc_install_get_info;
```

C++

```
Tisc_install_get_info isc_install_get_info;
```

File

IBIntf

Description

This is variable IBIntf.isc_install_get_info.

10.2.41 IBIntf.isc_install_get_message Variable

Pascal

```
isc_install_get_message: Tisc_install_get_message;
```

C++

```
Tisc_install_get_message isc_install_get_message;
```

File

IBIntf

Description

This is variable IBIntf.isc_install_get_message.

10.2.42 IBIntf.isc_install_load_external_text Variable

Pascal

```
isc_install_load_external_text: Tisc_install_load_external_text;
```

C++

```
Tisc_install_load_external_text isc_install_load_external_text;
```

File

IBIntf

Description

This is variable IBIntf.isc_install_load_external_text.

10.2.43 IBIntf.isc_install_precheck Variable

Pascal

```
isc_install_precheck: Tisc_install_precheck;
```

C++

```
Tisc_install_precheck isc_install_precheck;
```

File

IBIntf

Description

This is variable IBIntf.isc_install_precheck.

10.2.44 IBIntf.isc_install_set_option Variable

Pascal

```
isc_install_set_option: Tisc_install_set_option;
```

C++

```
Tisc_install_set_option isc_install_set_option;
```

File

IBIntf

Description

This is variable IBIntf.isc_install_set_option.

10.2.45 IBIntf.isc_install_unset_option Variable

Pascal

```
isc_install_unset_option: Tisc_install_unset_option;
```

C++

```
Tisc_install_unset_option isc_install_unset_option;
```

File

IBIntf

Description

This is variable IBIntf.isc_install_unset_option.

10.2.46 IBIntf.isc_interprete Variable

Pascal

```
isc_interprete: Tisc_interprete;
```

C++

```
Tisc_interprete isc_interprete;
```

File

IBIntf

Description

This is variable IBIntf.isc_interprete.

10.2.47 IBIntf.isc_modify_user Variable

Pascal

```
isc_modify_user: Tisc_modify_user;
```

C++

```
Tisc_modify_user isc_modify_user;
```

File

IBIntf

Description

This is variable IBIntf.isc_modify_user.

10.2.48 IBIntf.isc_open_blob2 Variable

Pascal

```
isc_open_blob2: Tisc_open_blob2;
```

C++

```
Tisc_open_blob2 isc_open_blob2;
```

File

IBIntf

Description

This is variable IBIntf.isc_open_blob2.

10.2.49 IBIntf.isc_prepare_transaction Variable

Pascal

```
isc_prepare_transaction: Tisc_prepare_transaction;
```

C++

```
Tisc_prepare_transaction isc_prepare_transaction;
```

File

IBIntf

Description

This is variable IBIntf.isc_prepare_transaction.

10.2.50 IBIntf.isc_prepare_transaction2 Variable

Pascal

```
isc_prepare_transaction2: Tisc_prepare_transaction2;
```

C++

```
Tisc_prepare_transaction2 isc_prepare_transaction2;
```

File

IBIntf

Description

This is variable IBIntf.isc_prepare_transaction2.

10.2.51 IBIntf.isc_put_segment Variable

Pascal

```
isc_put_segment: Tisc_put_segment;
```

C++

```
Tisc_put_segment isc_put_segment;
```

File

IBIntf

Description

This is variable IBIntf.isc_put_segment.

10.2.52 IBIntf.isc_que_events Variable

Pascal

```
isc_que_events: Tisc_que_events;
```

C++

```
Tisc_que_events isc_que_events;
```

File

IBIntf

Description

This is variable IBIntf.isc_que_events.

10.2.53 IBIntf.isc_rollback_retaining Variable

Pascal

```
isc_rollback_retaining: Tisc_rollback_retaining;
```

C++

```
Tisc_rollback_retaining isc_rollback_retaining;
```

File

IBIntf

Description

This is variable IBIntf.isc_rollback_retaining.

10.2.54 IBIntf.isc_rollback_transaction Variable

Pascal

```
isc_rollback_transaction: Tisc_rollback_transaction;
```

C++

```
Tisc_rollback_transaction isc_rollback_transaction;
```

File

IBIntf

Description

This is variable IBIntf.isc_rollback_transaction.

10.2.55 IBIntf.isc_service_attach Variable

Pascal

```
isc_service_attach: Tisc_service_attach;
```

C++

```
Tisc_service_attach isc_service_attach;
```

File

IBIntf

Description

This is variable IBIntf.isc_service_attach.

10.2.56 IBIntf.isc_service_detach Variable

Pascal

```
isc_service_detach: Tisc_service_detach;
```

C++

```
Tisc_service_detach isc_service_detach;
```

File

IBIntf

Description

This is variable IBIntf.isc_service_detach.

10.2.57 IBIntf.isc_service_query Variable

Pascal

```
isc_service_query: Tisc_service_query;
```

C++

```
Tisc_service_query isc_service_query;
```

File

IBIntf

Description

This is variable IBIntf.isc_service_query.

10.2.58 IBIntf.isc_service_start Variable

Pascal

```
isc_service_start: Tisc_service_start;
```

C++

```
Tisc_service_start isc_service_start;
```

File

IBIntf

Description

This is variable IBIntf.isc_service_start.

10.2.59 IBIntf.isc_sql_interprete Variable

Pascal

```
isc_sql_interprete: Tisc_sql_interprete;
```

C++

```
Tisc_sql_interprete isc_sql_interprete;
```

File

IBIntf

Description

This is variable IBIntf.isc_sql_interprete.

10.2.60 IBIntf.isc_sqlcode Variable

Pascal

```
isc_sqlcode: Tisc_sqlcode;
```

C++

```
Tisc_sqlcode isc_sqlcode;
```

File

IBIntf

Description

This is variable IBIntf.isc_sqlcode.

10.2.61 IBIntf.isc_start_multiple Variable

Pascal

```
isc_start_multiple: Tisc_start_multiple;
```

C++

```
Tisc_start_multiple isc_start_multiple;
```

File

IBIntf

Description

This is variable IBIntf.isc_start_multiple.

10.2.62 IBIntf.isc_uninstall_execute Variable

Pascal

```
isc_uninstall_execute: Tisc_uninstall_execute;
```

C++

```
Tisc_uninstall_execute isc_uninstall_execute;
```

File

IBIntf

Description

This is variable IBIntf.isc_uninstall_execute.

10.2.63 IBIntf.isc_uninstall_precheck Variable

Pascal

```
isc_uninstall_precheck: Tisc_uninstall_precheck;
```

C++

```
Tisc_uninstall_precheck isc_uninstall_precheck;
```

File

IBIntf

Description

This is variable IBIntf.isc_uninstall_precheck.

10.2.64 IBIntf.isc_vax_integer Variable

Pascal

```
isc_vax_integer: Tisc_vax_integer;
```

C++

```
Tisc_vax_integer isc_vax_integer;
```

File

IBIntf

Description

This is variable IBIntf.isc_vax_integer.

11 IBQuery Namespace

This is namespace IBQuery.

Topics

Name	Description
Classes (see page 376)	The following table lists classes in this documentation.

Classes

	Name	Description
	TIBQuery (see page 376)	TIBQuery executes an InterBase SQL (see page 392) statement.

11.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBQuery (see page 376)	TIBQuery executes an InterBase SQL (see page 392) statement.

11.1.1 TIBQuery Class

TIBQuery executes an InterBase SQL (see page 392) statement.

Class Hierarchy

Pascal

```
TIBQuery = class(TIBCustomDataSet);
```

C++

```
class TIBQuery : public TIBCustomDataSet;
```

File

IBQuery

Description

Use TIBQuery to access one or more tables in a database using SQL (see page 392) statements. The TIBQuery component provides a read-only dataset, and adapts well to the InterBase client/server environment. To update the result set that TIBQuery represents, use this component in conjunction with a TIBUpdateSQL component.

Query components are useful because they can

- Access more than one table at a time (called a 'join' in SQL (see page 392)).
- Automatically access a subset of rows and columns in its underlying table(s), rather than always returning all rows and columns.

Note: TIBQuery is of particular importance to the development of scalable database applications. If there is any chance that an application built to run against local databases will be scaled to a remote SQL (see page 392) database server in the future, use TIBQuery components from the start to ensure easier scaling later.

See Also

Types of Datasets

Getting Started with InterBase Express

11.1.1.1 TIBQuery Members

The following tables list the members exposed by TIBQuery.

Protected Methods

	Name	Description
	ActivateConnection (see page 69)	This is ActivateConnection, a member of class TIBCustomDataSet.
	ActivateTransaction (see page 69)	This is ActivateTransaction, a member of class TIBCustomDataSet.
V	AllocRecordBuffer (see page 69)	This is AllocRecordBuffer, a member of class TIBCustomDataSet.
	CheckDatasetClosed (see page 71)	This is CheckDatasetClosed, a member of class TIBCustomDataSet.
	CheckDatasetOpen (see page 71)	This is CheckDatasetOpen, a member of class TIBCustomDataSet.
	CheckNotUniDirectional (see page 71)	This is CheckNotUniDirectional, a member of class TIBCustomDataSet.
V	ClearCalcFields (see page 71)	This is ClearCalcFields, a member of class TIBCustomDataSet.
	ConstraintsStored (see page 72)	This is ConstraintsStored, a member of class TIBCustomDataSet.
V	CreateFields (see page 72)	This is CreateFields, a member of class TIBCustomDataSet.
	DeactivateTransaction (see page 73)	This is DeactivateTransaction, a member of class TIBCustomDataSet.
V	DefineProperties (see page 382)	This is DefineProperties, a member of class TIBQuery.
V	Disconnect (see page 383)	This is Disconnect, a member of class TIBQuery.
V	DoBeforeDelete (see page 73)	This is DoBeforeDelete, a member of class TIBCustomDataSet.
V	DoBeforeEdit (see page 73)	This is DoBeforeEdit, a member of class TIBCustomDataSet.
V	DoBeforeInsert (see page 74)	This is DoBeforeInsert, a member of class TIBCustomDataSet.
V	DoOnNewRecord (see page 74)	This is DoOnNewRecord, a member of class TIBCustomDataSet.
V	FreeRecordBuffer (see page 75)	This is FreeRecordBuffer, a member of class TIBCustomDataSet.
	GenerateQueryForLiveUpdate (see page 383)	This is GenerateQueryForLiveUpdate, a member of class TIBQuery.
	GetActiveBuf (see page 75)	This is GetActiveBuf, a member of class TIBCustomDataSet.
V	GetBookmarkData (see page 75)	This is GetBookmarkData, a member of class TIBCustomDataSet.
V	GetBookmarkFlag (see page 75)	This is GetBookmarkFlag, a member of class TIBCustomDataSet.
V	GetCanModify (see page 75)	This is GetCanModify, a member of class TIBCustomDataSet.
V	GetDataSource (see page 76)	This is GetDataSource, a member of class TIBCustomDataSet.
V	GetFieldClass (see page 76)	This is GetFieldClass, a member of class TIBCustomDataSet.
	GetParamsCount (see page 384)	This is GetParamsCount, a member of class TIBQuery.
V	GetRecNo (see page 77)	This is GetRecNo, a member of class TIBCustomDataSet.
V	GetRecord (see page 77)	This is GetRecord, a member of class TIBCustomDataSet.
V	GetRecordCount (see page 78)	This is GetRecordCount, a member of class TIBCustomDataSet.
V	GetRecordSize (see page 78)	This is GetRecordSize, a member of class TIBCustomDataSet.
V	InitFieldDefs (see page 384)	This is InitFieldDefs, a member of class TIBQuery.
V	InitRecord (see page 78)	This is InitRecord, a member of class TIBCustomDataSet.
V	InternalAddRecord (see page 78)	This is InternalAddRecord, a member of class TIBCustomDataSet.
V	InternalBatchInput (see page 79)	This is InternalBatchInput, a member of class TIBCustomDataSet.
V	InternalBatchOutput (see page 79)	This is InternalBatchOutput, a member of class TIBCustomDataSet.
V	InternalCancel (see page 79)	This is InternalCancel, a member of class TIBCustomDataSet.
V	InternalClose (see page 79)	This is InternalClose, a member of class TIBCustomDataSet.
V	InternalDelete (see page 79)	This is InternalDelete, a member of class TIBCustomDataSet.
V	InternalExecQuery (see page 80)	This is InternalExecQuery, a member of class TIBCustomDataSet.
V	InternalFirst (see page 80)	This is InternalFirst, a member of class TIBCustomDataSet.
V	InternalGotoBookmark (see page 80)	This is InternalGotoBookmark, a member of class TIBCustomDataSet.
V	InternalHandleException (see page 80)	This is InternalHandleException, a member of class TIBCustomDataSet.
V	InternalInitFieldDefs (see page 81)	This is InternalInitFieldDefs, a member of class TIBCustomDataSet.
V	InternalInitRecord (see page 81)	This is InternalInitRecord, a member of class TIBCustomDataSet.
V	InternalInsert (see page 81)	This is InternalInsert, a member of class TIBCustomDataSet.
V	InternalLast (see page 81)	This is InternalLast, a member of class TIBCustomDataSet.
V	InternalOpen (see page 384)	This is InternalOpen, a member of class TIBQuery.
V	InternalPost (see page 82)	This is InternalPost, a member of class TIBCustomDataSet.
V	InternalPrepare (see page 82)	This is InternalPrepare, a member of class TIBCustomDataSet.
V	InternalRefresh (see page 82)	This is InternalRefresh, a member of class TIBCustomDataSet.
V	InternalRefreshRow (see page 82)	This is InternalRefreshRow, a member of class TIBCustomDataSet.
V	InternalSetParamsFromCursor (see page 83)	This is InternalSetParamsFromCursor, a member of class TIBCustomDataSet.
V	InternalSetToRecord (see page 83)	This is InternalSetToRecord, a member of class TIBCustomDataSet.

	InternalUnPrepare (see page 83)	This is InternalUnPrepare, a member of class TIBCustomDataSet.
	IsCursorOpen (see page 83)	This is IsCursorOpen, a member of class TIBCustomDataSet.
	Loaded (see page 84)	This is Loaded, a member of class TIBCustomDataSet.
	PSEndTransaction (see page 86)	This is PSEndTransaction, a member of class TIBCustomDataSet.
	PSExecute (see page 385)	This is PSExecute, a member of class TIBQuery.
	PSExecuteStatement (see page 86)	This is PSExecuteStatement, a member of class TIBCustomDataSet.
	PSGetParams (see page 385)	This is PSGetParams, a member of class TIBQuery.
	PSGetQuoteChar (see page 87)	This is PSGetQuoteChar, a member of class TIBCustomDataSet.
	PSGetTableName (see page 385)	This is PSGetTableName, a member of class TIBQuery.
	PSGetUpdateException (see page 87)	This is PSGetUpdateException, a member of class TIBCustomDataSet.
	PSInTransaction (see page 87)	This is PSInTransaction, a member of class TIBCustomDataSet.
	PSIsSQLBased (see page 88)	This is PSIsSQLBased, a member of class TIBCustomDataSet.
	PSIsSQLSupported (see page 88)	This is PSIsSQLSupported, a member of class TIBCustomDataSet.
	PSReset (see page 88)	This is PSReset, a member of class TIBCustomDataSet.
	PSSetCommandText (see page 386)	This is PSSetCommandText, a member of class TIBQuery.
	PSSetParams (see page 386)	This is PSSetParams, a member of class TIBQuery.
	PSStartTransaction (see page 88)	This is PSStartTransaction, a member of class TIBCustomDataSet.
	PSUpdateRecord (see page 88)	This is PSUpdateRecord, a member of class TIBCustomDataSet.
	ReQuery (see page 89)	This is ReQuery, a member of class TIBCustomDataSet.
	SetActive (see page 89)	This is SetActive, a member of class TIBCustomDataSet.
	SetBookmarkData (see page 90)	This is SetBookmarkData, a member of class TIBCustomDataSet.
	SetBookmarkFlag (see page 90)	This is SetBookmarkFlag, a member of class TIBCustomDataSet.
	SetCachedUpdates (see page 90)	This is SetCachedUpdates, a member of class TIBCustomDataSet.
	SetDataSource (see page 90)	This is SetDataSource, a member of class TIBCustomDataSet.
	SetFieldData (see page 90)	This is the overview for the SetFieldData method overload.
	SetFiltered (see page 386)	This is SetFiltered, a member of class TIBQuery.
	SetRecNo (see page 91)	This is SetRecNo, a member of class TIBCustomDataSet.

Protected Properties

	Name	Description
	DeleteSQL (see page 99)	Holds the SQL statement used to delete rows from the dataset.
	InsertSQL (see page 100)	Holds the SQL statement used to insert rows into the dataset.
R	InternalPrepared (see page 100)	Indicates whether the dataset has prepared its queries for execution.
	ModifySQL (see page 101)	Provides the ability to access the SQL object encapsulating the ModifySQL statement.
R	QDelete (see page 103)	Provides the ability to directly access the SQL object encapsulating the DeleteSQL (see page 99) statement.
R	QInsert (see page 104)	Provides the ability to directly access the SQL object encapsulating the InsertSQL (see page 100) statement.
R	QModify (see page 104)	Provides the ability to directly access the SQL object encapsulating the ModifySQL (see page 101) statement.
R	QRefresh (see page 104)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL (see page 105) statement.
R	QSelect (see page 104)	Provides the ability to directly access the SQL object encapsulating the SelectSQL (see page 105) statement.
	RefreshSQL (see page 105)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.
	SelectSQL (see page 105)	Provides the ability to directly access the SQL object encapsulating the SelectSQL statement.
R	SelectStmtHandle (see page 105)	Gets the handle for the SELECT statement.
R	SQLParams (see page 106)	Specifies the parameters to use for any parameterized queries.
	UpdateMode (see page 107)	Indicates how records are located when updating records.

Public Methods

	Name	Description
	ApplyUpdates (see page 69)	Writes a dataset's pending cached updates to the database.

	BatchInput (see page 382)	Executes the parameterized query in SQL (see page 392) for input in the referenced input object.
	BatchOutput (see page 382)	Outputs the selected query in SQL (see page 392) to the referenced OutputObject.
V	BookmarkValid (see page 70)	Indicates the validity of a specified bookmark.
	CachedUpdateStatus (see page 70)	Returns the status of the cached updates.
	CancelUpdates (see page 70)	Clears all pending cached updates from the cache.
V	CompareBookmarks (see page 71)	Compares two bookmarks.
V	Create (see page 381)	Creates an instance of a query component.
V	CreateBlobStream (see page 72)	Returns a TIBDSBlobStream (see page 138) object for reading or writing the data in a specified blob field.
	Current (see page 73)	Returns the descriptor area that holds information about the current query.
V	Destroy (see page 381)	Destroys the instance of a query.
	ExecSQL (see page 383)	Executes the SQL (see page 392) statement for the query.
	FetchAll (see page 74)	Retrieves all records from the current cursor position to the end of the file and stores them locally.
V	GetCurrentRecord (see page 76)	Retrieves the current record into a buffer.
V	GetDetailLinkFields (see page 383)	Fills lists with the master and detail fields of the link.
V	GetFieldData (see page 76)	This is the overview for the GetFieldData method overload.
V	IsSequenced (see page 83)	IsSequenced indicates whether sequence numbers are available for database records.
V	Locate (see page 84)	Searches the dataset for a specified record and makes that record the current record.
	LocateNext (see page 85)	Searches the dataset for the record after a specified record and makes that record the current record.
V	Lookup (see page 85)	Retrieves field values from a record that matches specified search values.
	ParamByName (see page 384)	Accesses parameter information based on a specified parameter name.
V	Post (see page 86)	Writes a modified record to the database server.
	Prepare (see page 385)	Sends a query to the server for optimization prior to execution.
	RecordModified (see page 89)	Sets the record to modified or unmodified.
	RevertRecord (see page 89)	Restores the current record in the dataset to an unmodified state when cached updates are enabled.
	SQLType (see page 91)	Returns the type of query to be executed.
	Undelete (see page 92)	Restores a record deleted from the dataset.
	UnPrepare (see page 386)	Frees the resources allocated for a previously prepared query.
V	UpdateStatus (see page 92)	Reports the update status for the current record.

Public Properties

	Name	Description
R	DBHandle (see page 99)	Specifies the database handle for the dataset.
	GenerateParamNames (see page 389)	Generates a list of parameter names for the query.
	LiveMode (see page 389)	Indicates what types of edits, if any, can be made to the data in the dataset.
R	ParamCount (see page 390)	Indicates the current number of parameters for the query.
R	Plan (see page 103)	Indicates the PLAN portion of the dataset's SELECT query.
	Prepared (see page 391)	Determines whether or not a query is prepared for execution.
R	RowsAffected (see page 391)	Returns the number of rows operated upon by the latest query execution.
	StatementType (see page 392)	Returns the statement type of the QSelect query.
R	StmtHandle (see page 392)	Identifies the statement handle for the query.
R	Text (see page 392)	Points to the actual text of the SQL (see page 392) query.
R	TRHandle (see page 106)	Specifies the transaction handle for the dataset.
	UpdateRecordTypes (see page 108)	Specifies the type of records visible in a dataset when cached updates are enabled.
R	UpdatesPending (see page 108)	Indicates whether the cached updates buffer contains records that are not yet applied.

Published Events

	Name	Description
	AfterDatabaseDisconnect (see page 394)	Occurs immediately after the connection to the associated database is closed.
	AfterTransactionEnd (see page 394)	Occurs immediately after a transaction that includes this dataset's query ends.
	BeforeDatabaseDisconnect (see page 394)	Occurs immediately before the connection to the associated database is closed.
	BeforeTransactionEnd (see page 395)	Occurs immediately before a transaction that includes this dataset's query ends.
	DatabaseFree (see page 395)	Occurs when the database associated with this dataset is freed.
	OnUpdateError (see page 110)	Occurs if an exception is generated when cached updates are applied to a database.
	OnUpdateRecord (see page 111)	Occurs when cached updates are applied to a record.
	TransactionFree (see page 395)	Occurs when the transaction that includes this dataset's query is freed.

Published Properties

	Name	Description
	Active (see page 387)	This is Active, a member of class TIBQuery.
	AfterCancel (see page 94)	This is AfterCancel, a member of class TIBCustomDataSet.
	AfterClose (see page 94)	This is AfterClose, a member of class TIBCustomDataSet.
	AfterDelete (see page 94)	This is AfterDelete, a member of class TIBCustomDataSet.
	AfterEdit (see page 94)	This is AfterEdit, a member of class TIBCustomDataSet.
	AfterInsert (see page 95)	This is AfterInsert, a member of class TIBCustomDataSet.
	AfterOpen (see page 95)	This is AfterOpen, a member of class TIBCustomDataSet.
	AfterPost (see page 95)	This is AfterPost, a member of class TIBCustomDataSet.
	AfterRefresh (see page 95)	This is AfterRefresh, a member of class TIBCustomDataSet.
	AfterScroll (see page 95)	This is AfterScroll, a member of class TIBCustomDataSet.
	AutoCalcFields (see page 96)	This is AutoCalcFields, a member of class TIBCustomDataSet.
	BeforeCancel (see page 96)	This is BeforeCancel, a member of class TIBCustomDataSet.
	BeforeClose (see page 96)	This is BeforeClose, a member of class TIBCustomDataSet.
	BeforeDelete (see page 96)	This is BeforeDelete, a member of class TIBCustomDataSet.
	BeforeEdit (see page 97)	This is BeforeEdit, a member of class TIBCustomDataSet.
	BeforeInsert (see page 97)	This is BeforeInsert, a member of class TIBCustomDataSet.
	BeforeOpen (see page 97)	This is BeforeOpen, a member of class TIBCustomDataSet.
	BeforePost (see page 97)	This is BeforePost, a member of class TIBCustomDataSet.
	BeforeRefresh (see page 97)	This is BeforeRefresh, a member of class TIBCustomDataSet.
	BeforeScroll (see page 98)	This is BeforeScroll, a member of class TIBCustomDataSet.
	BufferChunks (see page 387)	Specifies the dataset buffer chunk size.
	CachedUpdates (see page 388)	Specifies whether cached updates are enabled for a dataset.
	Constraints (see page 388)	This is Constraints, a member of class TIBQuery.
	Database (see page 99)	Identifies the database component for which this dataset represents one or more tables.
	DataSource (see page 388)	This is DataSource, a member of class TIBQuery.
	Filtered (see page 389)	Specifies whether filtering is active for a dataset.
	ForcedRefresh (see page 99)	Specifies whether the dataset should refresh its data every time it posts a record.
	GeneratorField (see page 389)	Specifies whether the dataset should refresh its data every time it posts a record.
	ObjectView (see page 101)	This is ObjectView, a member of class TIBCustomDataSet.
	OnCalcFields (see page 101)	This is OnCalcFields, a member of class TIBCustomDataSet.
	OnDeleteError (see page 102)	This is OnDeleteError, a member of class TIBCustomDataSet.
	OnEditError (see page 102)	This is OnEditError, a member of class TIBCustomDataSet.
	OnFilterRecord (see page 390)	This is OnFilterRecord, a member of class TIBQuery.
	OnNewRecord (see page 102)	This is OnNewRecord, a member of class TIBCustomDataSet.
	OnPostError (see page 102)	This is OnPostError, a member of class TIBCustomDataSet.
	ParamCheck (see page 390)	Specifies whether the parameter list for a query is regenerated if the SQL property changes at runtime.
	Params (see page 391)	Contains the parameters for a query's SQL (see page 392) statement.
	SQL (see page 392)	Contains the text of the SQL statement to execute for the query.
	Transaction (see page 106)	Identifies the transaction under which the query executes.

	UniDirectional (see page 393)	Determines whether bidirectional cursors are enabled for the dataset.
	UpdateObject (see page 393)	Specifies the update object component used to update a read-only result set.

11.1.1.2 TIBQuery.Create Constructor

Creates an instance of a query component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBQuery(TComponent * AOwner);
```

Description

Call Create to instantiate a query at runtime. Query components placed in forms or data modules at design time are created automatically.

Create calls its inherited Create constructor, creates an empty SQL (see page 392) statement list, creates an empty parameter list, sets the OnChange event handler for the SQL (see page 392) statement list, sets the ParamCheck (see page 390) property to True, and sets the RowsAffected (see page 391) property to -1.

11.1.1.3 TIBQuery.Destroy Destructor

Destroys the instance of a query.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBQuery();
```

Description

Do not call Destroy directly. Instead call Free to verify that the query is not already freed before calling Destroy. Destroy disconnects from the server, frees the SQL (see page 392) statement list and the parameter list, and then calls its inherited destructor.

11.1.1.4 TIBQuery Methods

The methods of the TIBQuery class are listed here.

Protected Methods

	Name	Description
	DefineProperties (see page 382)	This is DefineProperties, a member of class TIBQuery.
	Disconnect (see page 383)	This is Disconnect, a member of class TIBQuery.
	GenerateQueryForLiveUpdate (see page 383)	This is GenerateQueryForLiveUpdate, a member of class TIBQuery.
	GetParamsCount (see page 384)	This is GetParamsCount, a member of class TIBQuery.
	InitFieldDefs (see page 384)	This is InitFieldDefs, a member of class TIBQuery.
	InternalOpen (see page 384)	This is InternalOpen, a member of class TIBQuery.
	PSExecute (see page 385)	This is PSExecute, a member of class TIBQuery.
	PSGetParams (see page 385)	This is PSGetParams, a member of class TIBQuery.
	PSGetTableName (see page 385)	This is PSGetTableName, a member of class TIBQuery.

	PSSetCommandText (see page 386)	This is PSSetCommandText, a member of class TIBQuery.
	PSSetParams (see page 386)	This is PSSetParams, a member of class TIBQuery.
	SetFiltered (see page 386)	This is SetFiltered, a member of class TIBQuery.

Public Methods

	Name	Description
	BatchInput (see page 382)	Executes the parameterized query in SQL (see page 392) for input in the referenced input object.
	BatchOutput (see page 382)	Outputs the selected query in SQL (see page 392) to the referenced OutputObject.
	ExecSQL (see page 383)	Executes the SQL (see page 392) statement for the query.
	GetDetailLinkFields (see page 383)	Fills lists with the master and detail fields of the link.
	ParamByName (see page 384)	Accesses parameter information based on a specified parameter name.
	Prepare (see page 385)	Sends a query to the server for optimization prior to execution.
	UnPrepare (see page 386)	Frees the resources allocated for a previously prepared query.

11.1.1.4.1 TIBQuery.BatchInput Method

Executes the parameterized query in SQL (see page 392) for input in the referenced input object.

Pascal

```
procedure BatchInput(InputObject: TIBBatchInput);
```

C++

```
__fastcall BatchInput(TIBBatchInput InputObject);
```

Description

Call BatchInput to execute the parameterized query in SQL (see page 392) for input in the referenced input object.

11.1.1.4.2 TIBQuery.BatchOutput Method

Outputs the selected query in SQL (see page 392) to the referenced OutputObject.

Pascal

```
procedure BatchOutput(OutputObject: TIBBatchOutput);
```

C++

```
__fastcall BatchOutput(TIBBatchOutput OutputObject);
```

Description

Call BatchOutput to output the selected query in SQL (see page 392) to the referenced OutputObject.

11.1.1.4.3 TIBQuery.DefineProperties Method

Pascal

```
procedure DefineProperties(Filer: TFiler); override;
```

C++

```
virtual __fastcall DefineProperties(TFiler Filer);
```

Description

This is DefineProperties, a member of class TIBQuery.

11.1.1.4.4 TIBQuery.Disconnect Method

Pascal

```
procedure Disconnect; override;
```

C++

```
virtual __fastcall Disconnect();
```

Description

This is Disconnect, a member of class TIBQuery.

11.1.1.4.5 TIBQuery.ExecSQL Method

Executes the SQL (see page 392) statement for the query.

Pascal

```
procedure ExecSQL;
```

C++

```
__fastcall ExecSQL();
```

Description

Call ExecSQL to execute the SQL (see page 392) statement currently assigned to the SQL (see page 392) property. Use ExecSQL to execute queries that do not return a cursor to data (such as INSERT, UPDATE, DELETE, and CREATE TABLE).

Note: For SELECT statements, call Open instead of ExecSQL.

ExecSQL prepares the statement in SQL (see page 392) property for execution if it has not already been prepared. To speed performance, an application should ordinarily call Prepare (see page 385) before calling ExecSQL for the first time.

11.1.1.4.6 TIBQuery.GenerateQueryForLiveUpdate Method

Pascal

```
function GenerateQueryForLiveUpdate: Boolean;
```

C++

```
__fastcall Boolean GenerateQueryForLiveUpdate();
```

Description

This is GenerateQueryForLiveUpdate, a member of class TIBQuery.

11.1.1.4.7 TIBQuery.GetDetailLinkFields Method

Fills lists with the master and detail fields of the link.

Pascal

```
procedure GetDetailLinkFields(MasterFields: TList; DetailFields: TList); override;
```

C++

```
virtual __fastcall GetDetailLinkFields(TList MasterFields, TList DetailFields);
```

Description

Creates two lists of TFields from the master-detail relationship between two tables; one containing the master fields, and the

other containing the detail fields.

11.1.1.4.8 TIBQuery.GetParamsCount Method

Pascal

```
function GetParamsCount: Word;
```

C++

```
__fastcall Word GetParamsCount();
```

Description

This is GetParamsCount, a member of class TIBQuery.

11.1.1.4.9 TIBQuery.InitFieldDefs Method

Pascal

```
procedure InitFieldDefs; override;
```

C++

```
virtual __fastcall InitFieldDefs();
```

Description

This is InitFieldDefs, a member of class TIBQuery.

11.1.1.4.10 TIBQuery.InternalOpen Method

Pascal

```
procedure InternalOpen; override;
```

C++

```
virtual __fastcall InternalOpen();
```

Description

This is InternalOpen, a member of class TIBQuery.

11.1.1.4.11 TIBQuery.ParamByName Method

Accesses parameter information based on a specified parameter name.

Pascal

```
function ParamByName(const Value: string): TParam;
```

C++

```
__fastcall TParam ParamByName(const UnicodeString Value);
```

Description

Call ParamByName to set or use parameter information for a specific parameter based on its name.

Value is the name of the parameter for which to retrieve information.

ParamByName is primarily used to set an parameter's value at runtime. For example, the following statement retrieves the current value of a parameter called 'Contact' into an edit box:

```
Edit1.Text := Query1.ParamByName('Contact').AsString;
```

```
Edit1->Text = Query1->ParamByName("Contact")->AsString;
```

11.1.1.4.12 TIBQuery.Prepare Method

Sends a query to the server for optimization prior to execution.

Pascal

```
procedure Prepare;
```

C++

```
__fastcall Prepare();
```

Description

Call Prepare to have the remote database server allocate resources for the query and to perform additional optimizations. Calling Prepare before executing a query improves application performance.

A query is automatically prepared before being executed. A query that was automatically prepared will then be automatically unprepared after it executes. When a query will be executed a number of times, an application should always explicitly prepare the query to avoid multiple and unnecessary prepares and unprepares.

Preparing a query consumes some database resources, so it is good practice for an application to unprepare a query once it is done using it. The UnPrepare (see page 386) method unprepares a query.

Note: When you change the text of a query at runtime, the query is automatically closed and unprepared.

11.1.1.4.13 TIBQuery.PSExecute Method

Pascal

```
procedure PSExecute; override;
```

C++

```
virtual __fastcall PSExecute();
```

Description

This is PSExecute, a member of class TIBQuery.

11.1.1.4.14 TIBQuery.PSGetParams Method

Pascal

```
function PSGetParams: TParams; override;
```

C++

```
virtual __fastcall TParams PSGetParams();
```

Description

This is PSGetParams, a member of class TIBQuery.

11.1.1.4.15 TIBQuery.PSGetTableName Method

Pascal

```
function PSGetTableName: string; override;
```

C++

```
virtual __fastcall UnicodeString PSGetTableName();
```

Description

This is PSGetTableName, a member of class TIBQuery.

11.1.1.4.16 TIBQuery.PSSetCommandText Method

Pascal

```
procedure PSSetCommandText(const CommandText: string); override;
```

C++

```
virtual __fastcall PSSetCommandText(const UnicodeString CommandText);
```

Description

This is PSSetCommandText, a member of class TIBQuery.

11.1.1.4.17 TIBQuery.PSSetParams Method

Pascal

```
procedure PSSetParams(AParams: TParams); override;
```

C++

```
virtual __fastcall PSSetParams(TParams AParams);
```

Description

This is PSSetParams, a member of class TIBQuery.

11.1.1.4.18 TIBQuery.SetFiltered Method

Pascal

```
procedure SetFiltered(Value: Boolean); override;
```

C++

```
virtual __fastcall SetFiltered(Boolean Value);
```

Description

This is SetFiltered, a member of class TIBQuery.

11.1.1.4.19 TIBQuery.UnPrepare Method

Frees the resources allocated for a previously prepared query.

Pascal

```
procedure UnPrepare;
```

C++

```
__fastcall UnPrepare();
```

Description

Call UnPrepare to free the resources allocated for a previously prepared query on the server and client sides.

Preparing a query consumes some database resources, so it is good practice for an application to unprepare a query once it is done using it. The UnPrepare method unprepares a query.

Note: When you change the text of a query at runtime, the query is automatically closed and unprepared.

11.1.1.5 TIBQuery Properties

The properties of the TIBQuery class are listed here.

Public Properties

	Name	Description
	GenerateParamNames (see page 389)	Generates a list of parameter names for the query.
	LiveMode (see page 389)	Indicates what types of edits, if any, can be made to the data in the dataset.
R	ParamCount (see page 390)	Indicates the current number of parameters for the query.
	Prepared (see page 391)	Determines whether or not a query is prepared for execution.
R	RowsAffected (see page 391)	Returns the number of rows operated upon by the latest query execution.
	StatementType (see page 392)	Returns the statement type of the QSelect query.
R	StmtHandle (see page 392)	Identifies the statement handle for the query.
R	Text (see page 392)	Points to the actual text of the SQL (see page 392) query.

Published Properties

	Name	Description
	Active (see page 387)	This is Active, a member of class TIBQuery.
	BufferChunks (see page 387)	Specifies the dataset buffer chunk size.
	CachedUpdates (see page 388)	Specifies whether cached updates are enabled for a dataset.
	Constraints (see page 388)	This is Constraints, a member of class TIBQuery.
	DataSource (see page 388)	This is DataSource, a member of class TIBQuery.
	Filtered (see page 389)	Specifies whether filtering is active for a dataset.
	GeneratorField (see page 389)	Specifies whether the dataset should refresh its data every time it posts a record.
	OnFilterRecord (see page 390)	This is OnFilterRecord, a member of class TIBQuery.
	ParamCheck (see page 390)	Specifies whether the parameter list for a query is regenerated if the SQL property changes at runtime.
	Params (see page 391)	Contains the parameters for a query's SQL (see page 392) statement.
	SQL (see page 392)	Contains the text of the SQL statement to execute for the query.
	UniDirectional (see page 393)	Determines whether bidirectional cursors are enabled for the dataset.
	UpdateObject (see page 393)	Specifies the update object component used to update a read-only result set.

11.1.1.5.1 TIBQuery.Active Property

Pascal

```
property Active;
```

C++

```
__property Active;
```

Description

This is Active, a member of class TIBQuery.

11.1.1.5.2 TIBQuery.BufferChunks Property

Specifies the dataset buffer chunk size.

Pascal

```
property BufferChunks: Integer;
```

C++

```
__property int BufferChunks;
```

Description

Use BufferChunks to get or set the number of records for which the dataset allocates buffer space at any time. When the dataset's buffer is full, trying to fetch an additional record causes the dataset to reallocate the buffer so that it has enough memory to hold an additional BufferChunks records.

Note: When UniDirectional is true, BufferChunks has no meaning.

11.1.1.5.3 TIBQuery.CachedUpdates Property

Specifies whether cached updates are enabled for a dataset.

Pascal

```
property CachedUpdates: Boolean;
```

C++

```
__property Boolean CachedUpdates;
```

Description

CachedUpdates enables or disables the use of cached updates for a dataset. If CachedUpdates is true, cached updates are enabled. If CachedUpdates is false, cached updates are disabled.

When cached updates are enabled, updates to a dataset (such as posting changes, inserting new records, or deleting records), are stored in an internal cache on the client machine instead of being written directly to the dataset's underlying database tables. When changes are complete, an application writes all cached changes to the database in the context of a single transaction.

Note: **Note**: Instead of using cached updates, applications can obtain the same benefits with greater control by using a client dataset.

11.1.1.5.4 TIBQuery.Constraints Property

Pascal

```
property Constraints;
```

C++

```
__property Constraints;
```

Description

This is Constraints, a member of class TIBQuery.

11.1.1.5.5 TIBQuery.DataSource Property

Pascal

```
property DataSource;
```

C++

```
__property DataSource;
```

Description

This is DataSource, a member of class TIBQuery.

11.1.1.5.6 TIBQuery.Filtered Property

Specifies whether filtering is active for a dataset.

Pascal

```
property Filtered;
```

C++

```
__property Filtered;
```

Description

Check Filtered to determine whether dataset filtering is in effect. If Filtered is true, then filtering is active. Otherwise Filtered is false. To apply filter conditions specified in the Filter property or the OnFilterRecord (see page 390) event handler, set Filtered to true. Filtered is set to false by default.

Note: When filtering is enabled, user edits to a record may mean that the record no longer meets a filter's test condition. The next time the record is retrieved from the dataset while the filter is in effect, the record may seem to disappear. If that happens, the next record that passes the filter condition becomes the current record.

11.1.1.5.7 TIBQuery.GenerateParamNames Property

Generates a list of parameter names for the query.

Pascal

```
property GenerateParamNames: Boolean;
```

C++

```
__property Boolean GenerateParamNames;
```

Description

Set GenerateParamNames to true to have the query generate a list of parameter names.

11.1.1.5.8 TIBQuery.GeneratorField Property

Specifies whether the dataset should refresh its data every time it posts a record.

Pascal

```
property GeneratorField: TIBGeneratorField;
```

C++

```
__property TIBGeneratorField GeneratorField;
```

Description

Use GeneratorField to specify a generator that automatically provides the value for one of the fields in the dataset. Use the properties of the TIBGeneratorField object to indicate which field's value should be set and how that value should be calculated.

Warning: Do not set the value of GeneratorField. To change the value of this property, change the properties of the TIBGeneratorField object that is the value of this property.

11.1.1.5.9 TIBQuery.LiveMode Property

Indicates what types of edits, if any, can be made to the data in the dataset.

Pascal

```
property LiveMode: TLiveModes;
```

C++

```
__property TLiveModes LiveMode;
```

Description

Read LiveMode to determine what types of edits should be allowed. LiveMode is set automatically to restrict the capabilities of the dataset to reflect the user's rights.

Note: For read-only datasets, such as TIBQuery, LiveMode is always an empty set.

11.1.1.5.10 TIBQuery.OnFilterRecord Property

Pascal

```
property OnFilterRecord;
```

C++

```
__property OnFilterRecord;
```

Description

This is OnFilterRecord, a member of class TIBQuery.

11.1.1.5.11 TIBQuery.ParamCheck Property

Specifies whether the parameter list for a query is regenerated if the SQL property changes at runtime.

Pascal

```
property ParamCheck: Boolean;
```

C++

```
__property Boolean ParamCheck;
```

Description

Set ParamCheck to specify whether the Params property is cleared and regenerated if an application modifies the SQL in this dataset's query. By default ParamCheck is true, meaning that the Params property is automatically regenerated at runtime. When ParamCheck is true, the proper number of parameters is guaranteed to be generated for the current SQL statement.

This property is useful for data definition language (DDL) statements that contain parameters as part of the DDL statement that are not parameters for the TIBCustomDataSet. For example, the DDL statement to create a stored procedure may contain parameter statements that are part of the stored procedure. Set ParamCheck to false to prevent these parameters from being mistaken for parameters of the TIBCustomDataSet executing the DDL statement.

11.1.1.5.12 TIBQuery.ParamCount Property

Indicates the current number of parameters for the query.

Pascal

```
property ParamCount: Word;
```

C++

```
__property Word ParamCount;
```

Description

Inspect ParamCount to determine how many parameters are in the Params (see page 391) property. If the ParamCheck (see page 390) property is true, ParamCount always corresponds to the number of actual parameters in the SQL (see page 392) statement for the query.

11.1.1.5.13 TIBQuery.Params Property

Contains the parameters for a query's SQL (see page 392) statement.

Pascal

```
property Params: TParams;
```

C++

```
__property TParams Params;
```

Description

Access Params at runtime to view and set parameter names, values, and data types dynamically (at design time use the collection editor for the Params property to set parameter information). Params is a zero-based array of TParams parameter records. Index specifies the array element to access.

Note: An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName (see page 384). ParamByName (see page 384) cannot, however, be used to change a parameter's data type or name.

11.1.1.5.14 TIBQuery.Prepared Property

Determines whether or not a query is prepared for execution.

Pascal

```
property Prepared: Boolean;
```

C++

```
__property Boolean Prepared;
```

Description

Examine Prepared to determine if a query is already prepared for execution. If Prepared is true, the query is prepared, and if Prepared is false, the query is not prepared. While a query need not be prepared before execution, execution performance is enhanced if the query is prepared beforehand, particularly if it is a parameterized query that is executed more than once using the same parameter values.

Note: An application can change the current setting of Prepared to prepare or unprepare a query. If Prepared is true, setting it to false calls the UnPrepare (see page 386) method to unprepare the query. If Prepared is false, setting it to true calls the Prepare (see page 385) method to prepare the query. Generally, however, it is better programming practice to call Prepare (see page 385) and UnPrepare (see page 386) directly. These methods automatically update the Prepared property.

11.1.1.5.15 TIBQuery.RowsAffected Property

Returns the number of rows operated upon by the latest query execution.

Pascal

```
property RowsAffected: Integer;
```

C++

```
__property int RowsAffected;
```

Description

Check RowsAffected to determine how many rows were updated or deleted by the last query operation. If RowsAffected is -1, the query did not update or delete any rows.

11.1.1.5.16 TIBQuery.SQL Property

Contains the text of the SQL statement to execute for the query.

Pascal

```
property SQL: TStrings;
```

C++

```
__property TStrings SQL;
```

Description

Use SQL to provide the SQL statement that a query component executes when its ExecSQL (see page 383) or Open method is called. At design time the SQL property can be edited by invoking the String List editor in the Object Inspector.

The SQL property may contain only one complete SQL statement at a time.

11.1.1.5.17 TIBQuery.StatementType Property

Returns the statement type of the QSelect query.

Pascal

```
property StatementType: TIBSQLTypes;
```

C++

```
__property TIBSQLTypes StatementType;
```

Description

Use StatementType to determine the statement type of a QSelect query.

11.1.1.5.18 TIBQuery.StmtHandle Property

Identifies the statement handle for the query.

Pascal

```
property StmtHandle: TISC_STMT_HANDLE;
```

C++

```
__property TISC_STMT_HANDLE StmtHandle;
```

Description

Retrieve StmtHandle if an application makes a direct call to the InterBase server, bypassing the methods of TIBQuery (see page 376). Some API calls require a statement handle as a parameter. Under all other circumstances an application does not need to access this property.

11.1.1.5.19 TIBQuery.Text Property

Points to the actual text of the SQL (see page 392) query.

Pascal

```
property Text: string;
```

C++

```
__property UnicodeString Text;
```

Description

Text is a read-only property that can be examined to determine the actual contents of SQL (see page 392) statement. For parameterized queries, Text contains the SQL (see page 392) statement with parameters replaced by the parameter substitution symbol (?) in place of actual parameter values.

In general there should be no need to examine the Text property. To access or change the SQL (see page 392) statement for the query, use the SQL (see page 392) property. To examine or modify parameters, use the Params (see page 391) property.

11.1.1.5.20 TIBQuery.UniDirectional Property

Determines whether bidirectional cursors are enabled for the dataset.

Pascal

```
property UniDirectional: Boolean;
```

C++

```
__property Boolean UniDirectional;
```

Description

Use UniDirectional to determine whether bidirectional cursors are enabled for this dataset.

Note: Do not confuse the UniDirectional property with the IsUniDirectional property inherited from TDataSet. IsUniDirectional indicates whether the dataset is a dbExpress dataset, which, in addition to using a unidirectional cursor, does not buffer records. UniDirectional is only concerned with whether the cursor permits backward or random navigation.

11.1.1.5.21 TIBQuery.UpdateObject Property

Specifies the update object component used to update a read-only result set.

Pascal

```
property UpdateObject: TIBDataSetUpdateObject;
```

C++

```
__property TIBDataSetUpdateObject UpdateObject;
```

Description

Use UpdateObject to specify the TIBDataSetUpdateObject component to use in an application that must be able to update a read-only result set.

In some cases, such as when using TIBQuery, the dataset does not automatically update records. In these cases, UpdateObject can be used to specify a TIBUpdateSQL component that performs updates as a separate transaction that is transparent to the application.

TIBDataSetUpdateObject is especially useful for complicated updates from an OnUpdateRecord event.

11.1.1.6 TIBQuery Events

The events of the TIBQuery class are listed here.

Published Events

	Name	Description
	AfterDatabaseDisconnect (see page 394)	Occurs immediately after the connection to the associated database is closed.
	AfterTransactionEnd (see page 394)	Occurs immediately after a transaction that includes this dataset's query ends.
	BeforeDatabaseDisconnect (see page 394)	Occurs immediately before the connection to the associated database is closed.
	BeforeTransactionEnd (see page 395)	Occurs immediately before a transaction that includes this dataset's query ends.
	DatabaseFree (see page 395)	Occurs when the database associated with this dataset is freed.
	TransactionFree (see page 395)	Occurs when the transaction that includes this dataset's query is freed.

11.1.1.6.1 TIBQuery.AfterDatabaseDisconnect Event

Occurs immediately after the connection to the associated database is closed.

Pascal

```
property AfterDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterDatabaseDisconnect;
```

Description

Write an AfterDatabaseDisconnect event handler to respond when the dataset closes its connection to the database specified by the Database property.

11.1.1.6.2 TIBQuery.AfterTransactionEnd Event

Occurs immediately after a transaction that includes this dataset's query ends.

Pascal

```
property AfterTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterTransactionEnd;
```

Description

Write an AfterTransactionEnd event handler to respond when the transaction specified by the Transaction property ends.

11.1.1.6.3 TIBQuery.BeforeDatabaseDisconnect Event

Occurs immediately before the connection to the associated database is closed.

Pascal

```
property BeforeDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeDatabaseDisconnect;
```

Description

Write a BeforeDatabaseDisconnect event handler to respond when the dataset is about to close its connection to the database

specified by the Database property.

11.1.1.6.4 TIBQuery.BeforeTransactionEnd Event

Occurs immediately before a transaction that includes this dataset's query ends.

Pascal

```
property BeforeTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeTransactionEnd;
```

Description

Write a BeforeTransactionEnd event handler to respond when the transaction specified by the Transaction property is about to end.

11.1.1.6.5 TIBQuery.DatabaseFree Event

Occurs when the database associated with this dataset is freed.

Pascal

```
property DatabaseFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent DatabaseFree;
```

Description

Write a DatabaseFree event handler to respond when the database specified by the Database property is freed.

11.1.1.6.6 TIBQuery.TransactionFree Event

Occurs when the transaction that includes this dataset's query is freed.

Pascal

```
property TransactionFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent TransactionFree;
```

Description

Write a TransactionFree event handler to respond when the transaction specified by the Transaction property is freed.

12 IBServices Namespace

This is namespace IBServices.

Topics

Name	Description
Classes (see page 399)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 477)	The following table lists structs, records, enums in this documentation.
Types (see page 486)	The following table lists types in this documentation.
Constants (see page 491)	The following table lists constants in this documentation.

Classes

	Name	Description
	TConfigFileData (see page 399)	This is class IBServices.TConfigFileData.
	TConfigParams (see page 401)	TConfigFileData (see page 399) and TConfigParams hold configuration information about an InterBase server
	TDatabaseInfo (see page 404)	TDatabaseInfo stores information about an InterBase database.
	TIBBackupRestoreService (see page 406)	TIBBackupRestoreService is the ancestor object from which TIBBackupService (see page 408) and TIBRestoreService (see page 439) are derived.
	TIBBackupService (see page 408)	TIBBackupService allows you to back up your database.
	TIBConfigService (see page 412)	TIBConfigService configures the database parameters
	TIBControlAndQueryService (see page 417)	TIBControlAndQueryService is the ancestor object from which the log, statistical, validation, security, and backup and restore IBService components descend.
	TIBControlService (see page 421)	TIBControlService is the ancestor object from which the structured and unstructured IBService components descend.
	TIBCustomService (see page 425)	TIBCustomService is the ancestor object from which all IBService components descend.
	TIBLicensingService (see page 433)	TIBLicensingService configures the licensing parameters
	TIBLogService (see page 437)	TIBLogService returns the contents of the interbase.log file from a server.
	TIBRestoreService (see page 439)	TIBRestoreService allows you to restore your database.
	TIBSecurityService (see page 443)	TIBSecurityService enables you to manage user access to the InterBase server.
	TIBServerProperties (see page 452)	TIBServerProperties returns database server information.
	TIBStatisticalService (see page 458)	TIBStatisticalService allows you to view database statistics.
	TIBValidationService (see page 461)	TIBValidationService enables you to validate your database and reconcile your database transactions.
	TLicenseInfo (see page 466)	TLicenseInfo stores information about licensed users.
	TLicenseMaskInfo (see page 468)	TLicenseMaskInfo indicates the software activation certificate options enabled on the server.
	TLimboTransactionInfo (see page 469)	TLimboTransactionInfo stores information about a limbo transaction.

	TUserInfo (see page 472)	TUserInfo stores information about an InterBase user for the security service.
	TVersionInfo (see page 475)	TVersionInfo represents version information about an InterBase server.

Constants

Name	Description
DefaultBufferSize (see page 491)	This is constant IBServices.DefaultBufferSize.
SPBConstantNames (see page 491)	This is constant IBServices.SPBConstantNames.
SPBConstantValues (see page 492)	This is constant IBServices.SPBConstantValues.
SPBPrefix (see page 492)	This is constant IBServices.SPBPrefix.

Structs, Records, Enums

	Name	Description
	TBackupOption (see page 477)	This is record IBServices.TBackupOption.
	TLicensingAction (see page 478)	TLicensingAction indicates whether to add or remove InterBase software activation certificates.
	TOutputBufferOption (see page 478)	This is record IBServices.TOutputBufferOption.
	TPropertyOption (see page 479)	This is record IBServices.TPropertyOption.
	TProtocol (see page 479)	TProtocol identifies a network protocol.
	TRestoreOption (see page 480)	This is record IBServices.TRestoreOption.
	TSecurityAction (see page 480)	TSecurityAction indicates what operation the InterBase security service should perform.
	TSecurityModifyParam (see page 481)	This is record IBServices.TSecurityModifyParam.
	TShutdownMode (see page 481)	TShutdownMode indicates what steps to take when shutting down the InterBase database.
	TStatOption (see page 482)	This is record IBServices.TStatOption.
	TTransactionAction (see page 482)	TTransactionAction indicates the action a transaction can take when it times out.
	TTransactionAdvise (see page 483)	This is record IBServices.TTransactionAdvise.
	TTransactionGlobalAction (see page 483)	TTransactionGlobalAction indicates the action to take on a set of transactions.
	TTransactionState (see page 484)	This is record IBServices.TTransactionState.
	TValidateOption (see page 485)	This is record IBServices.TValidateOption.

Types

Name	Description
TBackupOptions (see page 486)	TBackupOption (see page 477) and TBackupOptions indicate how TIBBackupService (see page 408) performs database backup.
TLoginEvent (see page 487)	This is type IBServices.TLoginEvent.
TPropertyOptions (see page 487)	TPropertyOption (see page 479) and TPropertyOptions indicate groups of InterBase server properties.
TRestoreOptions (see page 488)	TRestoreOption (see page 480) and TRestoreOptions indicate what operations take place when restoring an InterBase database.
TSecurityModifyParams (see page 489)	This is type IBServices.TSecurityModifyParams.
TStatOptions (see page 489)	TStatOption (see page 482) and TStatOptions indicate the different statistical information available about a database.
TValidateOptions (see page 490)	TValidateOption (see page 485) and TValidateOptions indicate what operations the validation service should perform when restoring a database.

12.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TConfigFileData (see page 399)	This is class IBServices.TConfigFileData.
	TConfigParams (see page 401)	TConfigFileData (see page 399) and TConfigParams hold configuration information about an InterBase server
	TDatabaseInfo (see page 404)	TDatabaseInfo stores information about an InterBase database.
	TIBBackupRestoreService (see page 406)	TIBBackupRestoreService is the ancestor object from which TIBBackupService (see page 408) and TIBRestoreService (see page 439) are derived.
	TIBBackupService (see page 408)	TIBBackupService allows you to back up your database.
	TIBConfigService (see page 412)	TIBConfigService configures the database parameters
	TIBControlAndQueryService (see page 417)	TIBControlAndQueryService is the ancestor object from which the log, statistical, validation, security, and backup and restore IBService components descend.
	TIBControlService (see page 421)	TIBControlService is the ancestor object from which the structured and unstructured IBService components descend.
	TIBCustomService (see page 425)	TIBCustomService is the ancestor object from which all IBService components descend.
	TIBLicensingService (see page 433)	TIBLicensingService configures the licensing parameters
	TIBLogService (see page 437)	TIBLogService returns the contents of the interbase.log file from a server.
	TIBRestoreService (see page 439)	TIBRestoreService allows you to restore your database.
	TIBSecurityService (see page 443)	TIBSecurityService enables you to manage user access to the InterBase server.
	TIBServerProperties (see page 452)	TIBServerProperties returns database server information.
	TIBStatisticalService (see page 458)	TIBStatisticalService allows you to view database statistics.
	TIBValidationService (see page 461)	TIBValidationService enables you to validate your database and reconcile your database transactions.
	TLicenseInfo (see page 466)	TLicenseInfo stores information about licensed users.
	TLicenseMaskInfo (see page 468)	TLicenseMaskInfo indicates the software activation certificate options enabled on the server.
	TLimboTransactionInfo (see page 469)	TLimboTransactionInfo stores information about a limbo transaction.
	TUserInfo (see page 472)	TUserInfo stores information about an InterBase user for the security service.
	TVersionInfo (see page 475)	TVersionInfo represents version information about an InterBase server.

12.1.1 TConfigFileData Class

Class Hierarchy

Pascal

```
TConfigFileData = class;
```

C++

```
class TConfigFileData;
```

File

IBServices

Description

This is class IBServices.TConfigFileData.

12.1.1.1 TConfigFileData Members

The following tables list the members exposed by TConfigFileData.

Public Fields

	Name	Description
	ConfigFileKey (see page 401)	This is ConfigFileKey, a member of class TConfigFileData.
	ConfigFileValue (see page 401)	This is ConfigFileValue, a member of class TConfigFileData.

Public Methods

	Name	Description
	Create (see page 400)	This is Create, a member of class TConfigFileData.
	Destroy (see page 400)	This is Destroy, a member of class TConfigFileData.

12.1.1.2 TConfigFileData.Create Constructor

Pascal

```
constructor Create;
```

C++

```
__fastcall TConfigFileData();
```

Description

This is Create, a member of class TConfigFileData.

12.1.1.3 TConfigFileData.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TConfigFileData();
```

Description

This is Destroy, a member of class TConfigFileData.

12.1.1.4 TConfigFileData Fields

The fields of the TConfigFileData class are listed here.

Public Fields

	Name	Description
	ConfigFileKey (see page 401)	This is ConfigFileKey, a member of class TConfigFileData.
	ConfigFileValue (see page 401)	This is ConfigFileValue, a member of class TConfigFileData.

12.1.1.4.1 TConfigFileData.ConfigFileKey Field

Pascal

```
ConfigFileKey: array of integer;
```

C++

```
array of integer ConfigFileKey;
```

Description

This is ConfigFileKey, a member of class TConfigFileData.

12.1.1.4.2 TConfigFileData.ConfigFileValue Field

Pascal

```
ConfigFileValue: array of integer;
```

C++

```
array of integer ConfigFileValue;
```

Description

This is ConfigFileValue, a member of class TConfigFileData.

12.1.2 TConfigParams Class

TConfigFileData (see page 399) and TConfigParams hold configuration information about an InterBase server

Class Hierarchy

Pascal

```
TConfigParams = class;
```

C++

```
class TConfigParams;
```

File

IBServices

Description

TConfigFileData (see page 399) stored information about an InterBase configuration file. It is one of the parameters of the TConfigParams type, which stores server configuration settings.

12.1.2.1 TConfigParams Members

The following tables list the members exposed by TConfigParams.

Public Fields

	Name	Description
	BaseLocation (see page 402)	This is BaseLocation, a member of class TConfigParams.

	ConfigFileData (see page 403)	This is ConfigFileData, a member of class TConfigParams.
	ConfigFileParams (see page 403)	This is ConfigFileParams, a member of class TConfigParams.
	LockFileLocation (see page 403)	This is LockFileLocation, a member of class TConfigParams.
	MessageFileLocation (see page 403)	This is MessageFileLocation, a member of class TConfigParams.
	SecurityDatabaseLocation (see page 404)	This is SecurityDatabaseLocation, a member of class TConfigParams.

Public Methods

	Name	Description
	Create (see page 402)	This is Create, a member of class TConfigParams.
V	Destroy (see page 402)	This is Destroy, a member of class TConfigParams.

12.1.2.2 TConfigParams.Create Constructor

Pascal

```
constructor Create;
```

C++

```
__fastcall TConfigParams();
```

Description

This is Create, a member of class TConfigParams.

12.1.2.3 TConfigParams.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TConfigParams();
```

Description

This is Destroy, a member of class TConfigParams.

12.1.2.4 TConfigParams Fields

The fields of the TConfigParams class are listed here.

Public Fields

	Name	Description
	BaseLocation (see page 402)	This is BaseLocation, a member of class TConfigParams.
	ConfigFileData (see page 403)	This is ConfigFileData, a member of class TConfigParams.
	ConfigFileParams (see page 403)	This is ConfigFileParams, a member of class TConfigParams.
	LockFileLocation (see page 403)	This is LockFileLocation, a member of class TConfigParams.
	MessageFileLocation (see page 403)	This is MessageFileLocation, a member of class TConfigParams.
	SecurityDatabaseLocation (see page 404)	This is SecurityDatabaseLocation, a member of class TConfigParams.

12.1.2.4.1 TConfigParams.BaseLocation Field

Pascal

```
BaseLocation: string;
```

C++

```
UnicodeString BaseLocation;
```

Description

This is BaseLocation, a member of class TConfigParams.

12.1.2.4.2 TConfigParams.ConfigFileData Field

Pascal

```
ConfigFileData: TConfigFileData;
```

C++

```
TConfigFileData ConfigFileData;
```

Description

This is ConfigFileData, a member of class TConfigParams.

12.1.2.4.3 TConfigParams.ConfigFileParams Field

Pascal

```
ConfigFileParams: array of string;
```

C++

```
array of string ConfigFileParams;
```

Description

This is ConfigFileParams, a member of class TConfigParams.

12.1.2.4.4 TConfigParams.LockFileLocation Field

Pascal

```
LockFileLocation: string;
```

C++

```
UnicodeString LockFileLocation;
```

Description

This is LockFileLocation, a member of class TConfigParams.

12.1.2.4.5 TConfigParams.MessageFileLocation Field

Pascal

```
MessageFileLocation: string;
```

C++

```
UnicodeString MessageFileLocation;
```

Description

This is MessageFileLocation, a member of class TConfigParams.

12.1.2.4.6 TConfigParams.SecurityDatabaseLocation Field

Pascal

```
SecurityDatabaseLocation: string;
```

C++

```
UnicodeString SecurityDatabaseLocation;
```

Description

This is SecurityDatabaseLocation, a member of class TConfigParams.

12.1.3 TDatabaseInfo Class

TDatabaseInfo stores information about an InterBase database.

Class Hierarchy

Pascal

```
TDatabaseInfo = class;
```

C++

```
class TDatabaseInfo;
```

File

IBServices

Description

TDatabaseInfo describes an InterBase database.

NoOfAttachments (see page 405) is the number attachments.

NoOfDatabases (see page 406) is the number of databases.

DbName (see page 405) is the name of the database.

12.1.3.1 TDatabaseInfo Members

The following tables list the members exposed by TDatabaseInfo.

Public Fields

	Name	Description
	DbName (see page 405)	This is DbName, a member of class TDatabaseInfo.
	NoOfAttachments (see page 405)	This is NoOfAttachments, a member of class TDatabaseInfo.
	NoOfDatabases (see page 406)	This is NoOfDatabases, a member of class TDatabaseInfo.

Public Methods

	Name	Description
	Create (see page 405)	This is Create, a member of class TDatabaseInfo.
	Destroy (see page 405)	This is Destroy, a member of class TDatabaseInfo.

12.1.3.2 TDatabaseInfo.Create Constructor

Pascal

```
constructor Create;
```

C++

```
__fastcall TDatabaseInfo();
```

Description

This is Create, a member of class TDatabaseInfo.

12.1.3.3 TDatabaseInfo.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TDatabaseInfo();
```

Description

This is Destroy, a member of class TDatabaseInfo.

12.1.3.4 TDatabaseInfo Fields

The fields of the TDatabaseInfo class are listed here.

Public Fields

	Name	Description
	DbName (see page 405)	This is DbName, a member of class TDatabaseInfo.
	NoOfAttachments (see page 405)	This is NoOfAttachments, a member of class TDatabaseInfo.
	NoOfDatabases (see page 406)	This is NoOfDatabases, a member of class TDatabaseInfo.

12.1.3.4.1 TDatabaseInfo.DbName Field

Pascal

```
DbName: array of string;
```

C++

```
array of string DbName;
```

Description

This is DbName, a member of class TDatabaseInfo.

12.1.3.4.2 TDatabaseInfo.NoOfAttachments Field

Pascal

```
NoOfAttachments: Integer;
```

C++

```
int NoOfAttachments;
```

Description

This is NoOfAttachments, a member of class TDatabaseInfo.

12.1.3.4.3 TDatabaseInfo.NoOfDatabases Field

Pascal

```
NoOfDatabases: Integer;
```

C++

```
int NoOfDatabases;
```

Description

This is NoOfDatabases, a member of class TDatabaseInfo.

12.1.4 TIBBackupRestoreService Class

TIBBackupRestoreService is the ancestor object from which TIBBackupService (see page 408) and TIBRestoreService (see page 439) are derived.

Class Hierarchy

Pascal

```
TIBBackupRestoreService = class(TIBControlAndQueryService);
```

C++

```
class TIBBackupRestoreService : public TIBControlAndQueryService;
```

File

IBServices

Description

TIBBackupRestoreService is the ancestor object from which TIBBackupService (see page 408) and TIBRestoreService (see page 439) are derived.

Note: You must install InterBase 6 to use this feature.

12.1.4.1 TIBBackupRestoreService Members

The following tables list the members exposed by TIBBackupRestoreService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.

	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
V	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.
	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 424)	This is SetServiceStartOptions, a member of class TIBControlService.

Protected Properties

	Name	Description
	Action (see page 420)	This is Action, a member of class TIBControlAndQueryService.
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Methods

	Name	Description
	Attach (see page 427)	Attaches to the database.
V	Create (see page 419)	Creates an instance of a TIBControlAndQueryService (see page 417).
V	Create (see page 422)	Creates an instance of TIBControlService (see page 421)
V	Create (see page 426)	Creates an instance of TIBCustomService (see page 425)
V	Destroy (see page 426)	Destroys the service.
	Detach (see page 428)	Detaches from the database.
	GetNextChunk (see page 419)	Gets the next chunk of data.
	GetNextLine (see page 420)	Starts the service.
V	ServiceStart (see page 423)	Starts the service.

Public Properties

	Name	Description
R	Eof (see page 421)	Indicates that the end of file has been reached.
R	Handle (see page 430)	Returns the database handle.
R	IsServiceRunning (see page 424)	Indicates whether the service is running.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	BufferSize (see page 420)	Sets or returns the buffer size.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.
	Verbose (see page 408)	Sets or returns the backup or restore in verbose mode.

12.1.4.2 TIBBackupRestoreService Properties

The properties of the TIBBackupRestoreService class are listed here.

Published Properties

	Name	Description
	Verbose (see page 408)	Sets or returns the backup or restore in verbose mode.

12.1.4.2.1 TIBBackupRestoreService.Verbose Property

Sets or returns the backup or restore in verbose mode.

Pascal

```
property Verbose: Boolean;
```

C++

```
__property Boolean Verbose;
```

Description

Use Verbose to set or return the backup or restore in verbose mode.

Note: You must install InterBase 6 to use this feature.

12.1.5 TIBBackupService Class

TIBBackupService allows you to back up your database.

Class Hierarchy

Pascal

```
TIBBackupService = class(TIBBackupRestoreService);
```

C++

```
class TIBBackupService : public TIBBackupRestoreService;
```

File

IBServices

Description

Use a TIBBackupService object to back up your database.

See Also

Getting Started with InterBase Express

12.1.5.1 TIBBackupService Members

The following tables list the members exposed by TIBBackupService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.

	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
V	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.
	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 410)	This is SetServiceStartOptions, a member of class TIBBackupService.

Protected Properties

	Name	Description
	Action (see page 420)	This is Action, a member of class TIBControlAndQueryService.
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Methods

	Name	Description
	Attach (see page 427)	Attaches to the database.
V	Create (see page 410)	Creates an instance of TIBBackupService (see page 408)
V	Destroy (see page 410)	Destroys the service.
	Detach (see page 428)	Detaches from the database.
	GetNextChunk (see page 419)	Gets the next chunk of data.
	GetNextLine (see page 420)	Starts the service.
V	ServiceStart (see page 423)	Starts the service.

Public Properties

	Name	Description
R	Eof (see page 421)	Indicates that the end of file has been reached.
R	Handle (see page 430)	Returns the database handle.
R	IsServiceRunning (see page 424)	Indicates whether the service is running.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	BackupFile (see page 411)	Sets or returns the backup file name.
	BlockingFactor (see page 411)	Sets or returns the blocking factor for the tape device.
	BufferSize (see page 420)	Sets or returns the buffer size.
	DatabaseName (see page 411)	Sets or returns the database name.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Options (see page 411)	Sets the options for the database backup.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.
	Verbose (see page 408)	Sets or returns the backup or restore in verbose mode.

12.1.5.2 TIBBackupService.Create Constructor

Creates an instance of TIBBackupService (see page 408)

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBBackupService(TComponent * AOwner);
```

Description

Call Create to create an instance of TIBBackupService (see page 408).

12.1.5.3 TIBBackupService.Destroy Destructor

Destroys the service.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBBackupService();
```

Description

Do not call Destroy directly in an application. Instead, an application should call Free. Free verifies that the service object has not already been freed before it calls Destroy.

12.1.5.4 TIBBackupService Methods

The methods of the TIBBackupService class are listed here.

Protected Methods

	Name	Description
	SetServiceStartOptions (see page 410)	This is SetServiceStartOptions, a member of class TIBBackupService.

12.1.5.4.1 TIBBackupService.SetServiceStartOptions Method

Pascal

```
procedure SetServiceStartOptions; override;
```

C++

```
virtual __fastcall SetServiceStartOptions();
```

Description

This is SetServiceStartOptions, a member of class TIBBackupService.

12.1.5.5 TIBBackupService Properties

The properties of the TIBBackupService class are listed here.

Published Properties

	Name	Description
	BackupFile (see page 411)	Sets or returns the backup file name.
	BlockingFactor (see page 411)	Sets or returns the blocking factor for the tape device.
	DatabaseName (see page 411)	Sets or returns the database name.
	Options (see page 411)	Sets the options for the database backup.

12.1.5.5.1 TIBBackupService.BackupFile Property

Sets or returns the backup file name.

Pascal

```
property BackupFile: TStrings;
```

C++

```
__property TStrings BackupFile;
```

Description

Use BackupFile to set or return the backup file name.

12.1.5.5.2 TIBBackupService.BlockingFactor Property

Sets or returns the blocking factor for the tape device.

Pascal

```
property BlockingFactor: Integer;
```

C++

```
__property int BlockingFactor;
```

Description

Use BlockingFactor to set the blocking factor for the tape device as an integer.

12.1.5.5.3 TIBBackupService.DatabaseName Property

Sets or returns the database name.

Pascal

```
property DatabaseName: string;
```

C++

```
__property UnicodeString DatabaseName;
```

Description

Use DatabaseName to set or return the database name.

12.1.5.5.4 TIBBackupService.Options Property

Sets the options for the database backup.

Pascal

```
property Options: TBackupOptions;
```

C++

```
__property TBackupOptions Options;
```

Description

Use Options to select the options for the database backup.

12.1.6 TIBConfigService Class

TIBConfigService configures the database parameters

Class Hierarchy

Pascal

```
TIBConfigService = class(TIBControlService);
```

C++

```
class TIBConfigService : public TIBControlService;
```

File

IBServices

Description

Use a TIBConfigService object to configure database parameters.

Note: You must install InterBase 6 to use this feature.

See Also

Getting Started with InterBase Express

12.1.6.1 TIBConfigService Members

The following tables list the members exposed by TIBConfigService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.
	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
V	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.
	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 424)	This is SetServiceStartOptions, a member of class TIBControlService.

Protected Properties

	Name	Description
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.

	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Methods

	Name	Description
	ActivateShadow (see page 414)	Activates the shadow file for database use.
	Attach (see page 427)	Attaches to the database.
	BringDatabaseOnline (see page 414)	Brings the database online.
V	Create (see page 422)	Creates an instance of TIBControlService (see page 421)
V	Create (see page 426)	Creates an instance of TIBCustomService (see page 425)
V	Destroy (see page 426)	Destroys the service.
	Detach (see page 428)	Detaches from the database.
V	ServiceStart (see page 414)	Starts the service.
	SetAsyncMode (see page 415)	Sets database writes to asynchronous mode.
	SetDBSqlDialect (see page 415)	Sets the SQL dialect for the database.
	SetPageBuffers (see page 415)	Sets the number of database page buffers.
	SetReadOnly (see page 416)	Sets the database to read only.
	SetReserveSpace (see page 416)	Sets reserve space on the data page for versioning.
	SetSweepInterval (see page 416)	Sets the sweep interval.
	ShutdownDatabase (see page 417)	Shuts down the database.

Public Properties

	Name	Description
R	Handle (see page 430)	Returns the database handle.
R	IsServiceRunning (see page 424)	Indicates whether the service is running.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	DatabaseName (see page 417)	This is DatabaseName, a member of class TIBConfigService.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.

12.1.6.2 TIBConfigService Methods

The methods of the TIBConfigService class are listed here.

Public Methods

	Name	Description
	ActivateShadow (see page 414)	Activates the shadow file for database use.
	BringDatabaseOnline (see page 414)	Brings the database online.
V	ServiceStart (see page 414)	Starts the service.
	SetAsyncMode (see page 415)	Sets database writes to asynchronous mode.
	SetDBSqlDialect (see page 415)	Sets the SQL dialect for the database.
	SetPageBuffers (see page 415)	Sets the number of database page buffers.

	SetReadOnly (see page 416)	Sets the database to read only.
	SetReserveSpace (see page 416)	Sets reserve space on the data page for versioning.
	SetSweepInterval (see page 416)	Sets the sweep interval.
	ShutdownDatabase (see page 417)	Shuts down the database.

12.1.6.2.1 TIBConfigService.ActivateShadow Method

Activates the shadow file for database use.

Pascal

```
procedure ActivateShadow;
```

C++

```
__fastcall ActivateShadow();
```

Description

Call ActivateShadow to activate the database shadow.

InterBase 6 lets you recover a database in case of disk failure, network failure, or accidental deletion of the database. This recovery method is known as disk shadowing, or simply shadowing. Before you can activate shadowing, you must first create a database shadow, as discussed in 'Shadowing' and 'Creating a shadow' in the InterBase 6 Operations Guide.

Note: You must install InterBase 6 to use this feature.

12.1.6.2.2 TIBConfigService.BringDatabaseOnline Method

Brings the database online.

Pascal

```
procedure BringDatabaseOnline;
```

C++

```
__fastcall BringDatabaseOnline();
```

Description

Call BringDatabaseOnline to bring a database online.

Note: You must install InterBase 6 to use this feature.

12.1.6.2.3 TIBConfigService.ServiceStart Method

Starts the service.

Pascal

```
procedure ServiceStart; override;
```

C++

```
virtual __fastcall ServiceStart();
```

Description

Call ServiceStart to start the service.

Note: You must install InterBase 6 to use this feature.

12.1.6.2.4 TIBConfigService.SetAsyncMode Method

Sets database writes to asynchronous mode.

Pascal

```
procedure SetAsyncMode(Value: Boolean);
```

C++

```
__fastcall SetAsyncMode(Boolean Value);
```

Description

Set SetAsyncMode to true to change the database write mode to asynchronous (buffered writes).

InterBase 6 allows you to write to databases in both synchronous and asynchronous modes. In synchronous mode, the database writes are forced. In asynchronous mode, the database writes are buffered.

For more information, refer to 'Forced writes vs. buffered writes' in the InterBase 6 Operations Guide.

Note: You must install InterBase 6 to use this feature.

12.1.6.2.5 TIBConfigService.SetDBSqlDialect Method

Sets the SQL dialect for the database.

Pascal

```
procedure SetDBSqlDialect(Value: Integer);
```

C++

```
__fastcall SetDBSqlDialect(int Value);
```

Description

Call SetDBSqlDialect to set the SQL dialect for the database. Valid integer values are 1, 2, and 3.

Note: You must install InterBase 6 to use this feature.

12.1.6.2.6 TIBConfigService.SetPageBuffers Method

Sets the number of database page buffers.

Pascal

```
procedure SetPageBuffers(Value: Integer);
```

C++

```
__fastcall SetPageBuffers(int Value);
```

Description

Call SetPageBuffers to set the number of database page buffers.

When a program establishes a connection to a database, InterBase allocates system memory to use as a private buffer. The buffers are used to store accessed database pages to speed performance. The number of buffers assigned determines how many simultaneous database pages it can have access to in the memory pool. Buffers remain assigned until a program finishes with a database.

For more information on page buffers, refer to 'Setting database cache buffers' in the InterBase 6 Programmer's Guide.

Note: You must install InterBase 6 to use this feature.

12.1.6.2.7 TIBConfigService.SetReadOnly Method

Sets the database to read only.

Pascal

```
procedure SetReadOnly(Value: Boolean);
```

C++

```
__fastcall SetReadOnly(Boolean Value);
```

Description

Call SetReadOnly to set the database to read only.

Access mode specifies the type of access a transaction has for the table it uses. There are two possible modes: read-only and read-write.

- Read-only specifies that a transaction can read data from a table, but cannot insert, update, or delete table data.
- Read-write specifies that a transaction can select, insert, update, and delete table data. This is the default setting if none is specified.

Tip: You should specify the transaction's access mode even if is read-write. It makes the application's source code easier to read and debug, because the program's intentions are clearly spelled out.

Note: You must install InterBase 6 to use this feature.

12.1.6.2.8 TIBConfigService.SetReserveSpace Method

Sets reserve space on the data page for versioning.

Pascal

```
procedure SetReserveSpace(Value: Boolean);
```

C++

```
__fastcall SetReserveSpace(Boolean Value);
```

Description

Set SetReserveSpace to true to reserve space on the data page for versioning.

Note: You must install InterBase 6 to use this feature.

12.1.6.2.9 TIBConfigService.SetSweepInterval Method

Sets the sweep interval.

Pascal

```
procedure SetSweepInterval(Value: Integer);
```

C++

```
__fastcall SetSweepInterval(int Value);
```

Description

Call SetSweepInterval set the sweep interval for the database. The sweep interval refers to the number of transactions between database sweeps.

To turn off database sweeps, set the sweep interval to 0.

Sweeping a database is a systematic way of removing outdated records from the database. Periodic sweeping keeps the database from getting too large. However, sweeping can also slow performance. For more information, refer to 'Setting the sweep interval' in the InterBase 6 Operations Guide.

Note: You must install InterBase 6 to use this feature.

12.1.6.2.10 TIBConfigService.ShutdownDatabase Method

Shuts down the database.

Pascal

```
procedure ShutdownDatabase(Options: TShutdownMode; Wait: Integer);
```

C++

```
__fastcall ShutdownDatabase(TShutdownMode Options, int Wait);
```

Description

Call ShutdownDatabase to shut down the database after a specified number of seconds.

Note: You must install InterBase 6 to use this feature.

12.1.6.3 TIBConfigService Properties

The properties of the TIBConfigService class are listed here.

Published Properties

	Name	Description
	DatabaseName (see page 417)	This is DatabaseName, a member of class TIBConfigService.

12.1.6.3.1 TIBConfigService.DatabaseName Property

Pascal

```
property DatabaseName: string;
```

C++

```
__property UnicodeString DatabaseName;
```

Description

This is DatabaseName, a member of class TIBConfigService.

12.1.7 TIBControlAndQueryService Class

TIBControlAndQueryService is the ancestor object from which the log, statistical, validation, security, and backup and restore IBService components descend.

Class Hierarchy

Pascal

```
TIBControlAndQueryService = class(TIBControlService);
```

C++

```
class TIBControlAndQueryService : public TIBControlService;
```

File

IBServices

Description

TIBControlAndQueryService is the ancestor object from which the log, statistical, validation, security, and backup and restore IBService components descend. The service is unstructured because the output is in an unformatted text file.

Note: You must install InterBase 6 to use this feature.

12.1.7.1 TIBControlAndQueryService Members

The following tables list the members exposed by TIBControlAndQueryService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.
	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
V	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.
	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 424)	This is SetServiceStartOptions, a member of class TIBControlService.

Protected Properties

	Name	Description
	Action (see page 420)	This is Action, a member of class TIBControlAndQueryService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Methods

	Name	Description
	Attach (see page 427)	Attaches to the database.
V	Create (see page 419)	Creates an instance of a TIBControlAndQueryService (see page 417).
V	Destroy (see page 426)	Destroys the service.
	Detach (see page 428)	Detaches from the database.
	GetNextChunk (see page 419)	Gets the next chunk of data.
	GetNextLine (see page 420)	Starts the service.
V	ServiceStart (see page 423)	Starts the service.

Public Properties

	Name	Description
R	Eof (see page 421)	Indicates that the end of file has been reached.
R	Handle (see page 430)	Returns the database handle.
R	IsServiceRunning (see page 424)	Indicates whether the service is running.

	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	BufferSize (see page 420)	Sets or returns the buffer size.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.

12.1.7.2 TIBControlAndQueryService.Create Constructor

Creates an instance of a TIBControlAndQueryService (see page 417).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBControlAndQueryService(TComponent * AOwner);
```

Description

Call Create to create an instance of a TIBControlAndQueryService (see page 417) component.

Note: You must install InterBase 6 to use this feature.

12.1.7.3 TIBControlAndQueryService Methods

The methods of the TIBControlAndQueryService class are listed here.

Public Methods

	Name	Description
	GetNextChunk (see page 419)	Gets the next chunk of data.
	GetNextLine (see page 420)	Starts the service.

12.1.7.3.1 TIBControlAndQueryService.GetNextChunk Method

Gets the next chunk of data.

Pascal

```
function GetNextChunk: String;
```

C++

```
__fastcall UnicodeString GetNextChunk();
```

Description

Call GetNextChunk to get the next chunk of data.

Note: You must install InterBase 6 to use this feature.

12.1.7.3.2 TIBControlAndQueryService.GetNextLine Method

Starts the service.

Pascal

```
function GetNextLine: String;
```

C++

```
__fastcall UnicodeString GetNextLine();
```

Description

Call GetNextLine to get the next line of data.

Note: You must install InterBase 6 to use this feature.

12.1.7.4 TIBControlAndQueryService Properties

The properties of the TIBControlAndQueryService class are listed here.

Protected Properties

	Name	Description
	Action (see page 420)	This is Action, a member of class TIBControlAndQueryService.

Public Properties

	Name	Description
R	Eof (see page 421)	Indicates that the end of file has been reached.

Published Properties

	Name	Description
	BufferSize (see page 420)	Sets or returns the buffer size.

12.1.7.4.1 TIBControlAndQueryService.Action Property

Pascal

```
property Action: Integer;
```

C++

```
__property int Action;
```

Description

This is Action, a member of class TIBControlAndQueryService.

12.1.7.4.2 TIBControlAndQueryService.BufferSize Property

Sets or returns the buffer size.

Pascal

```
property BufferSize: Integer;
```

C++

```
__property int BufferSize;
```

Description

Use the BufferSize property to set or return the buffer size.

Note: You must install InterBase 6 to use this feature.

12.1.7.4.3 TIBControlAndQueryService.Eof Property

Indicates that the end of file has been reached.

Pascal

```
property Eof: boolean;
```

C++

```
__property boolean Eof;
```

Description

Use the Eof property to determine whether the end of the file has been reached.

Note: You must install InterBase 6 to use this feature.

12.1.8 TIBControlService Class

TIBControlService is the ancestor object from which the structured and unstructured IBService components descend.

Class Hierarchy

Pascal

```
TIBControlService = class(TIBCustomService);
```

C++

```
class TIBControlService : public TIBCustomService;
```

File

IBServices

Description

TIBControlService is the ancestor object from which the structured and unstructured IBService components descend.

Note: You must install InterBase 6 to use this feature.

12.1.8.1 TIBControlService Members

The following tables list the members exposed by TIBControlService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.

	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
V	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.
	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 424)	This is SetServiceStartOptions, a member of class TIBControlService.

Protected Properties

	Name	Description
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Methods

	Name	Description
	Attach (see page 427)	Attaches to the database.
V	Create (see page 422)	Creates an instance of TIBControlService (see page 421)
V	Destroy (see page 426)	Destroys the service.
	Detach (see page 428)	Detaches from the database.
V	ServiceStart (see page 423)	Starts the service.

Public Properties

	Name	Description
R	Handle (see page 430)	Returns the database handle.
R	IsServiceRunning (see page 424)	Indicates whether the service is running.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.

12.1.8.2 TIBControlService.Create Constructor

Creates an instance of TIBControlService (see page 421)

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBControlService(TComponent * AOwner);
```

Description

Call Create to create an instance of TIBControlService (see page 421).

Note: You must install InterBase 6 to use this feature.

12.1.8.3 TIBControlService Methods

The methods of the TIBControlService class are listed here.

Protected Methods

	Name	Description
	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 424)	This is SetServiceStartOptions, a member of class TIBControlService.

Public Methods

	Name	Description
V	ServiceStart (see page 423)	Starts the service.

12.1.8.3.1 TIBControlService.InternalServiceStart Method

Pascal

```
procedure InternalServiceStart;
```

C++

```
__fastcall InternalServiceStart();
```

Description

This is InternalServiceStart, a member of class TIBControlService.

12.1.8.3.2 TIBControlService.ServiceStart Method

Starts the service.

Pascal

```
procedure ServiceStart; virtual;
```

C++

```
virtual __fastcall ServiceStart();
```

Description

Call ServiceStart to start the service.

Note: You must install InterBase 6 to use this feature.

12.1.8.3.3 ServiceStartAddParam Method

This is the overview for the ServiceStartAddParam method overload.

Overload List

	Name	Description
	TIBControlService.ServiceStartAddParam (Integer, Integer) (see page 424)	This is ServiceStartAddParam, a member of class TIBControlService.

	TIBControlService.ServiceStartAddParam (string, Integer) (see page 424)	This is ServiceStartAddParam, a member of class TIBControlService.

12.1.8.3.3.1 TIBControlService.ServiceStartAddParam Method (Integer, Integer)

Pascal

```
procedure ServiceStartAddParam(Value: Integer; param: Integer); overload;
```

C++

```
__fastcall ServiceStartAddParam(int Value, int param);
```

Description

This is ServiceStartAddParam, a member of class TIBControlService.

12.1.8.3.3.2 TIBControlService.ServiceStartAddParam Method (string, Integer)

Pascal

```
procedure ServiceStartAddParam(Value: string; param: Integer); overload;
```

C++

```
__fastcall ServiceStartAddParam(UnicodeString Value, int param);
```

Description

This is ServiceStartAddParam, a member of class TIBControlService.

12.1.8.3.4 TIBControlService.SetServiceStartOptions Method

Pascal

```
procedure SetServiceStartOptions; virtual;
```

C++

```
virtual __fastcall SetServiceStartOptions();
```

Description

This is SetServiceStartOptions, a member of class TIBControlService.

12.1.8.4 TIBControlService Properties

The properties of the TIBControlService class are listed here.

Protected Properties

	Name	Description
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Properties

	Name	Description
R	IsServiceRunning (see page 424)	Indicates whether the service is running.

12.1.8.4.1 TIBControlService.IsServiceRunning Property

Indicates whether the service is running.

Pascal

```
property IsServiceRunning: Boolean;
```

C++

```
__property Boolean IsServiceRunning;
```

Description

Use the IsServiceRunning property to check whether the service is running.

Note: You must install InterBase 6 to use this feature.

12.1.8.4.2 TIBControlService.ServiceStartParams Property

Pascal

```
property ServiceStartParams: String;
```

C++

```
__property UnicodeString ServiceStartParams;
```

Description

This is ServiceStartParams, a member of class TIBControlService.

12.1.9 TIBCustomService Class

TIBCustomService is the ancestor object from which all IBService components descend.

Class Hierarchy

Pascal

```
TIBCustomService = class(TComponent);
```

C++

```
class TIBCustomService : public TComponent;
```

File

IBServices

Description

TIBCustomService is the ancestor object from which all IBService components descend.

12.1.9.1 TIBCustomService Members

The following tables list the members exposed by TIBCustomService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.

	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.

Protected Properties

	Name	Description
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.

Public Methods

	Name	Description
	Attach (see page 427)	Attaches to the database.
	Create (see page 426)	Creates an instance of TIBCustomService (see page 425)
	Destroy (see page 426)	Destroys the service.
	Detach (see page 428)	Detaches from the database.

Public Properties

	Name	Description
	Handle (see page 430)	Returns the database handle.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.

12.1.9.2 TIBCustomService.Create Constructor

Creates an instance of TIBCustomService (see page 425)

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBCustomService(TComponent * AOwner);
```

Description

Call Create to create an instance of TIBCustomService (see page 425).

12.1.9.3 TIBCustomService.Destroy Destructor

Destroys the service.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBCustomService();
```

Description

Do not call Destroy directly in an application. Instead, an application should call Free. Free verifies that the service object has not already been freed before it calls Destroy.

12.1.9.4 TIBCustomService Methods

The methods of the TIBCustomService class are listed here.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.
	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.

Public Methods

	Name	Description
	Attach (see page 427)	Attaches to the database.
	Detach (see page 428)	Detaches from the database.

12.1.9.4.1 TIBCustomService.Attach Method

Attaches to the database.

Pascal

```
procedure Attach;
```

C++

```
__fastcall Attach();
```

Description

Call Attach to attach to the database.

12.1.9.4.2 TIBCustomService.CheckActive Method

Pascal

```
procedure CheckActive;
```

C++

```
__fastcall CheckActive();
```

Description

This is CheckActive, a member of class TIBCustomService.

12.1.9.4.3 TIBCustomService.CheckInactive Method

Pascal

```
procedure CheckInactive;
```

C++

```
__fastcall CheckInactive();
```

Description

This is CheckInactive, a member of class TIBCustomService.

12.1.9.4.4 TIBCustomService.Detach Method

Detaches from the database.

Pascal

```
procedure Detach;
```

C++

```
__fastcall Detach();
```

Description

Call Detach to detach from the database.

12.1.9.4.5 TIBCustomService.InternalServiceQuery Method

Pascal

```
procedure InternalServiceQuery;
```

C++

```
__fastcall InternalServiceQuery();
```

Description

This is InternalServiceQuery, a member of class TIBCustomService.

12.1.9.4.6 TIBCustomService.Loaded Method

Initializes the component after the form file has been read into memory.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

Do not call the protected Loaded method. The streaming system calls this method after it loads the component's form from a stream.

When the streaming system loads a form or data module from its form file, it first constructs the form component by calling its constructor, then reads its property values from the form file. After reading all the property values for all the components, the streaming system calls the Loaded methods of each component in the order the components were created. This gives the components a chance to initialize any data that depends on the values of other components or other parts of itself.

Note: All references to sibling components are resolved by the time Loaded is called. Loaded is the first place that sibling pointers can be used after being streamed in.

As implemented in TComponent, Loaded clears the csLoading flag in the ComponentState property, indicating that the component is no longer loading.

Warning: Loaded may be called multiple times on inherited forms. It is called every time a level of inheritance is streamed in. Do not allocate memory in an overridden Loaded method without first checking that the memory has not been allocated in a previous call.

See Also

ComponentState

Create

TReader

12.1.9.4.7 TIBCustomService.Login Method

Pascal

```
function Login: Boolean;
```

C++

```
__fastcall Boolean Login();
```

Description

This is Login, a member of class TIBCustomService.

12.1.9.5 TIBCustomService Properties

The properties of the TIBCustomService class are listed here.

Protected Properties

	Name	Description
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.

Public Properties

	Name	Description
R	Handle (see page 430)	Returns the database handle.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.

12.1.9.5.1 TIBCustomService.Active Property

Sets the active state of the service.

Pascal

```
property Active: Boolean;
```

C++

```
__property Boolean Active;
```

Description

Use the Active property to set the service to active (true) or the default, inactive (false).

12.1.9.5.2 TIBCustomService.BufferSize Property

Pascal

```
property BufferSize: Integer;
```

C++

```
__property int BufferSize;
```

Description

This is BufferSize, a member of class TIBCustomService.

12.1.9.5.3 TIBCustomService.Handle Property

Returns the database handle.

Pascal

```
property Handle: TISC_SVC_HANDLE;
```

C++

```
__property TISC_SVC_HANDLE Handle;
```

Description

Use the Handle property to return the database handle.

12.1.9.5.4 TIBCustomService.LoginPrompt Property

Turns on and off login prompt display.

Pascal

```
property LoginPrompt: Boolean;
```

C++

```
__property Boolean LoginPrompt;
```

Description

Use the LoginPrompt**property to display (or not display) a login prompt before attaching to a database.**

12.1.9.5.5 TIBCustomService.OutputBuffer Property

Pascal

```
property OutputBuffer: PChar;
```

C++

```
__property const char * OutputBuffer;
```

Description

This is OutputBuffer, a member of class TIBCustomService.

12.1.9.5.6 TIBCustomService.OutputBufferOption Property

Pascal

```
property OutputBufferOption: TOutputBufferOption;
```

C++

```
__property TOutputBufferOption OutputBufferOption;
```

Description

This is OutputBufferOption, a member of class TIBCustomService.

12.1.9.5.7 TIBCustomService.Params Property

Sets or returns the database parameters

Pascal

```
property Params: TStrings;
```

C++

```
__property TStrings Params;
```

Description

Use the Params property to set or return database parameters.

12.1.9.5.8 TIBCustomService.Protocol Property

Sets or returns the network protocol.

Pascal

```
property Protocol: TProtocol;
```

C++

```
__property TProtocol Protocol;
```

Description

Use Protocol to select the network protocol.

12.1.9.5.9 TIBCustomService.ServerName Property

Sets the name of the server on which the services are to be run.

Pascal

```
property ServerName: string;
```

C++

```
__property UnicodeString ServerName;
```

Description

Use ServerName to set the name of the server on which the services are to be run.

12.1.9.5.10 TIBCustomService.ServiceParamBySPB Property

Returns and sets SPB parameters.

Pascal

```
property ServiceParamBySPB [const Idx: Integer]: String;
```

C++

```
__property UnicodeString ServiceParamBySPB[int const Idx];
```

Description

Use ParamBySPB to inspect and set SPB parameters.

For example DBParamBySPB[isc_SPB_user_name] sets and inspects the user name.

12.1.9.5.11 TIBCustomService.ServiceQueryParams Property

Pascal

```
property ServiceQueryParams: String;
```

C++

```
__property UnicodeString ServiceQueryParams;
```

Description

This is ServiceQueryParams, a member of class TIBCustomService.

12.1.9.5.12 TIBCustomService.TraceFlags Property

Pascal

```
property TraceFlags: TTraceFlags;
```

C++

```
__property TTraceFlags TraceFlags;
```

Description

This is TraceFlags, a member of class TIBCustomService.

12.1.9.6 TIBCustomService Events

The events of the TIBCustomService class are listed here.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

12.1.9.6.1 TIBCustomService.OnAttach Event

Occurs when the database is attached.

Pascal

```
property OnAttach: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnAttach;
```

Description

Write an OnAttach event handler to take specific actions when a database is attached. If an exception is raised in this event, the database is not attached.

12.1.9.6.2 TIBCustomService.OnLogin Event

Occurs during database login.

Pascal

```
property OnLogin: TLoginEvent;
```

C++

```
__property TLoginEvent OnLogin;
```

Description

Write an OnLogin event handler to take specific actions on database login.

12.1.10 TIBLicensingService Class

TIBLicensingService configures the licensing parameters

Class Hierarchy

Pascal

```
TIBLicensingService = class(TIBControlService);
```

C++

```
class TIBLicensingService : public TIBControlService;
```

File

IBServices

Description

Use a TIBLicensingService component to add or remove InterBase software activation certificates.

Note: You must install InterBase 6 to use this feature.

See Also

Getting Started with InterBase Express

12.1.10.1 TIBLicensingService Members

The following tables list the members exposed by TIBLicensingService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.
	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
V	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.
	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 436)	This is SetServiceStartOptions, a member of class TIBLicensingService.

Protected Properties

	Name	Description
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Methods

	Name	Description
	AddLicense (see page 435)	Adds an InterBase software activation certificate.
	Attach (see page 427)	Attaches to the database.
V	Create (see page 422)	Creates an instance of TIBControlService (see page 421)
V	Create (see page 426)	Creates an instance of TIBCustomService (see page 425)
V	Destroy (see page 426)	Destroys the service.
	Detach (see page 428)	Detaches from the database.
	RemoveLicense (see page 435)	Removes an InterBase software activation certificate.
V	ServiceStart (see page 423)	Starts the service.

Public Properties

	Name	Description
R	Handle (see page 430)	Returns the database handle.
R	IsServiceRunning (see page 424)	Indicates whether the service is running.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Action (see page 436)	Indictes the licensing action.

	Active (see page 430)	Sets the active state of the service.
	ID (see page 436)	Sets or returns the license identification.
	Key (see page 437)	Sets or returns the license key.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.

12.1.10.2 TIBLicensingService Methods

The methods of the TIBLicensingService class are listed here.

Protected Methods

	Name	Description
	SetServiceStartOptions (see page 436)	This is SetServiceStartOptions, a member of class TIBLicensingService.

Public Methods

	Name	Description
	AddLicense (see page 435)	Adds an InterBase software activation certificate.
	RemoveLicense (see page 435)	Removes an InterBase software activation certificate.

12.1.10.2.1 TIBLicensingService.AddLicense Method

Adds an InterBase software activation certificate.

Pascal

```
procedure AddLicense;
```

C++

```
__fastcall AddLicense();
```

Description

Call AddLicense along to add an InterBase software activation certificate. The Key (see page 437) and ID (see page 436) properties must be supplied.

Note: You must install InterBase 6 to use this feature.

12.1.10.2.2 TIBLicensingService.RemoveLicense Method

Removes an InterBase software activation certificate.

Pascal

```
procedure RemoveLicense;
```

C++

```
__fastcall RemoveLicense();
```

Description

Call RemoveLicense to remove an InterBase software activation certificate.

Note: You must install InterBase 6 to use this feature.

12.1.10.2.3 TIBLicensingService.SetServiceStartOptions Method

Pascal

```
procedure SetServiceStartOptions; override;
```

C++

```
virtual __fastcall SetServiceStartOptions();
```

Description

This is SetServiceStartOptions, a member of class TIBLicensingService.

12.1.10.3 TIBLicensingService Properties

The properties of the TIBLicensingService class are listed here.

Published Properties

	Name	Description
	Action (see page 436)	Indictes the licensing action.
	ID (see page 436)	Sets or returns the license identification.
	Key (see page 437)	Sets or returns the license key.

12.1.10.3.1 TIBLicensingService.Action Property

Indictes the licensing action.

Pascal

```
property Action: TLicensingAction;
```

C++

```
__property TLicensingAction Action;
```

Description

Use Action to indicate what action the licensing service should take. This causes the service to call the AddLicense (see page 435) or RemoveLicense (see page 435) method, depending on whether the software activation certificate is added or removed.

Note: You must install InterBase 6 to use this feature.

12.1.10.3.2 TIBLicensingService.ID Property

Sets or returns the license identification.

Pascal

```
property ID: String;
```

C++

```
__property UnicodeString ID;
```

Description

Use ID to set or return the license identification.

Note: You must install InterBase 6 to use this feature.

12.1.10.3.3 TIBLicensingService.Key Property

Sets or returns the license key.

Pascal

```
property Key: String;
```

C++

```
__property UnicodeString Key;
```

Description

Use Key to set or return the license key.

Note: You must install InterBase 6 to use this feature.

12.1.11 TIBLogService Class

TIBLogService returns the contents of the interbase.log file from a server.

Class Hierarchy

Pascal

```
TIBLogService = class(TIBControlAndQueryService);
```

C++

```
class TIBLogService : public TIBControlAndQueryService;
```

File

IBServices

Description

Use a TIBLogService object to return the contents of the interbase.log file from a server.

Note: You must install InterBase 6 to use this feature.

See Also

Getting Started with InterBase Express

12.1.11.1 TIBLogService Members

The following tables list the members exposed by TIBLogService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.
	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
V	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.

	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 439)	This is SetServiceStartOptions, a member of class TIBLogService.

Protected Properties

	Name	Description
	Action (see page 420)	This is Action, a member of class TIBControlAndQueryService.
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Methods

	Name	Description
	Attach (see page 427)	Attaches to the database.
V	Create (see page 419)	Creates an instance of a TIBControlAndQueryService (see page 417).
V	Create (see page 422)	Creates an instance of TIBControlService (see page 421)
V	Create (see page 426)	Creates an instance of TIBCustomService (see page 425)
V	Destroy (see page 426)	Destroys the service.
	Detach (see page 428)	Detaches from the database.
	GetNextChunk (see page 419)	Gets the next chunk of data.
	GetNextLine (see page 420)	Starts the service.
V	ServiceStart (see page 423)	Starts the service.

Public Properties

	Name	Description
R	Eof (see page 421)	Indicates that the end of file has been reached.
R	Handle (see page 430)	Returns the database handle.
R	IsServiceRunning (see page 424)	Indicates whether the service is running.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	BufferSize (see page 420)	Sets or returns the buffer size.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.

12.1.11.2 TIBLogService Methods

The methods of the TIBLogService class are listed here.

Protected Methods

	Name	Description
V	SetServiceStartOptions (see page 439)	This is SetServiceStartOptions, a member of class TIBLogService.

12.1.11.2.1 TIBLogService.SetServiceStartOptions Method

Pascal

```
procedure SetServiceStartOptions; override;
```

C++

```
virtual __fastcall SetServiceStartOptions();
```

Description

This is SetServiceStartOptions, a member of class TIBLogService.

12.1.12 TIBRestoreService Class

TIBRestoreService allows you to restore your database.

Class Hierarchy

Pascal

```
TIBRestoreService = class(TIBBackupRestoreService);
```

C++

```
class TIBRestoreService : public TIBBackupRestoreService;
```

File

IBServices

Description

Use a TIBRestoreService object to restore your database.

Note: You must install InterBase 6 to use this feature.

See Also

Getting Started with InterBase Express

12.1.12.1 TIBRestoreService Members

The following tables list the members exposed by TIBRestoreService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.
	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
V	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.
	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 441)	This is SetServiceStartOptions, a member of class TIBRestoreService.

Protected Properties

	Name	Description
	Action (see page 420)	This is Action, a member of class TIBControlAndQueryService.
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Methods

	Name	Description
	Attach (see page 427)	Attaches to the database.
V	Create (see page 440)	Creates an instance of TIBRestoreService (see page 439)
V	Destroy (see page 441)	Destroys the service.
	Detach (see page 428)	Detaches from the database.
	GetNextChunk (see page 419)	Gets the next chunk of data.
	GetNextLine (see page 420)	Starts the service.
V	ServiceStart (see page 423)	Starts the service.

Public Properties

	Name	Description
R	Eof (see page 421)	Indicates that the end of file has been reached.
R	Handle (see page 430)	Returns the database handle.
R	IsServiceRunning (see page 424)	Indicates whether the service is running.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	BackupFile (see page 442)	Sets or returns the backup file name.
	BufferSize (see page 420)	Sets or returns the buffer size.
	DatabaseName (see page 442)	Sets or returns the database name.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Options (see page 442)	Sets the options for the database restore.
	PageBuffers (see page 442)	Sets or returns page buffer size.
	PageSize (see page 443)	Sets or returns the page size.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.
	Verbose (see page 408)	Sets or returns the backup or restore in verbose mode.

12.1.12.2 TIBRestoreService.Create Constructor

Creates an instance of TIBRestoreService (see page 439)

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBRestoreService(TComponent * AOwner);
```

Description

Call Create to create an instance of TIBRestoreService (see page 439).

Note: You must install InterBase 6 to use this feature.

12.1.12.3 TIBRestoreService.Destroy Destructor

Destroys the service.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBRestoreService();
```

Description

Do not call Destroy directly in an application. Instead, an application should call Free. Free verifies that the service object has not already been freed before it calls Destroy.

Note: You must install InterBase 6 to use this feature.

12.1.12.4 TIBRestoreService Methods

The methods of the TIBRestoreService class are listed here.

Protected Methods

	Name	Description
	SetServiceStartOptions (see page 441)	This is SetServiceStartOptions, a member of class TIBRestoreService.

12.1.12.4.1 TIBRestoreService.SetServiceStartOptions Method

Pascal

```
procedure SetServiceStartOptions; override;
```

C++

```
virtual __fastcall SetServiceStartOptions();
```

Description

This is SetServiceStartOptions, a member of class TIBRestoreService.

12.1.12.5 TIBRestoreService Properties

The properties of the TIBRestoreService class are listed here.

Published Properties

	Name	Description
	BackupFile (see page 442)	Sets or returns the backup file name.

	DatabaseName (see page 442)	Sets or returns the database name.
	Options (see page 442)	Sets the options for the database restore.
	PageBuffers (see page 442)	Sets or returns page buffer size.
	PageSize (see page 443)	Sets or returns the page size.

12.1.12.5.1 TIBRestoreService.BackupFile Property

Sets or returns the backup file name.

Pascal

```
property BackupFile: TStrings;
```

C++

```
__property TStrings BackupFile;
```

Description

Use BackupFile to set or return the backup file name.

Note: You must install InterBase 6 to use this feature.

12.1.12.5.2 TIBRestoreService.DatabaseName Property

Sets or returns the database name.

Pascal

```
property DatabaseName: TStrings;
```

C++

```
__property TStrings DatabaseName;
```

Description

Use DatabaseName to set or return the database name.

Note: You must install InterBase 6 to use this feature.

12.1.12.5.3 TIBRestoreService.Options Property

Sets the options for the database restore.

Pascal

```
property Options: TRestoreOptions;
```

C++

```
__property TRestoreOptions Options;
```

Description

Use Options to select the options for the database backup.

Note: You must install InterBase 6 to use this feature.

12.1.12.5.4 TIBRestoreService.PageBuffers Property

Sets or returns page buffer size.

Pascal

```
property PageBuffers: Integer;
```

C++

```
__property int PageBuffers;
```

Description

Use PageBuffers to set or return the page buffer size in kilobytes.

Note: You must install InterBase 6 to use this feature.

12.1.12.5.5 TIBRestoreService.PageSize Property

Sets or returns the page size.

Pascal

```
property PageSize: Integer;
```

C++

```
__property int PageSize;
```

Description

Use PageSize to set or return the page size in kilobytes.

Note: You must install InterBase 6 to use this feature.

12.1.13 TIBSecurityService Class

TIBSecurityService enables you to manage user access to the InterBase server.

Class Hierarchy

Pascal

```
TIBSecurityService = class(TIBControlAndQueryService);
```

C++

```
class TIBSecurityService : public TIBControlAndQueryService;
```

File

IBServices

Description

Use a TIBSecurityService object to manage user access to the InterBase server.

Note: You must install InterBase 6 to use this feature.

See Also

Getting Started with InterBase Express

12.1.13.1 TIBSecurityService Members

The following tables list the members exposed by TIBSecurityService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.
	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
V	Loaded (see page 447)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.
	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 448)	This is SetServiceStartOptions, a member of class TIBSecurityService.

Protected Properties

	Name	Description
	Action (see page 420)	This is Action, a member of class TIBControlAndQueryService.
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Methods

	Name	Description
	AddUser (see page 446)	Adds a user to the user table.
	Attach (see page 427)	Attaches to the database.
V	Create (see page 445)	Creates an instance of TIBSecurityService (see page 443).
	DeleteUser (see page 446)	Deletes a user from the user table.
V	Destroy (see page 445)	Destroys the service.
	Detach (see page 428)	Detaches from the database.
	DisplayUser (see page 446)	Fetches user information.
	DisplayUsers (see page 447)	Fetches information for all users.
	GetNextChunk (see page 419)	Gets the next chunk of data.
	GetNextLine (see page 420)	Starts the service.
	ModifyUser (see page 448)	Modifies user information.
V	ServiceStart (see page 423)	Starts the service.

Public Properties

	Name	Description
R	Eof (see page 421)	Indicates that the end of file has been reached.
R	Handle (see page 430)	Returns the database handle.
R	IsServiceRunning (see page 424)	Indicates whether the service is running.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.
R	UserInfo (see page 451)	Provides information about a user.
R	UserInfoCount (see page 451)	Returns the number of elements in the TUserInfo (see page 472) array.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	BufferSize (see page 420)	Sets or returns the buffer size.
	FirstName (see page 449)	Sets or returns the user's first name.
	GroupID (see page 449)	Sets or returns the user's group ID number.
	LastName (see page 449)	Sets or returns the user's last name.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	MiddleName (see page 449)	Sets or returns the user's middle name.
	Params (see page 431)	Sets or returns the database parameters
	Password (see page 450)	Sets the user's password.
	Protocol (see page 431)	Sets or returns the network protocol.
	SecurityAction (see page 450)	Specifies the security action.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	SQlRole (see page 450)	The role name used to connect to the isc4.gdb security database.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.
	UserID (see page 450)	Sets or returns the user's ID number.
	UserName (see page 451)	Sets or returns the user's login name.

12.1.13.2 TIBSecurityService.Create Constructor

Creates an instance of TIBSecurityService (see page 443).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBSecurityService(TComponent * AOwner);
```

Description

Call Create to create an instance of TIBSecurityService (see page 443).

Note: You must install InterBase 6 to use this feature.

12.1.13.3 TIBSecurityService.Destroy Destructor

Destroys the service.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBSecurityService();
```

Description

Do not call Destroy directly in an application. Instead, an application should call Free. Free verifies that the service object has not already been freed before it calls Destroy.

Note: You must install InterBase 6 to use this feature.

12.1.13.4 TIBSecurityService Methods

The methods of the TIBSecurityService class are listed here.

Protected Methods

	Name	Description
	Loaded (see page 447)	Initializes the component after the form file has been read into memory.
	SetServiceStartOptions (see page 448)	This is SetServiceStartOptions, a member of class TIBSecurityService.

Public Methods

	Name	Description
	AddUser (see page 446)	Adds a user to the user table.
	DeleteUser (see page 446)	Deletes a user from the user table.
	DisplayUser (see page 446)	Fetches user information.
	DisplayUsers (see page 447)	Fetches information for all users.
	ModifyUser (see page 448)	Modifies user information.

12.1.13.4.1 TIBSecurityService.AddUser Method

Adds a user to the user table.

Pascal

```
procedure AddUser;
```

C++

```
__fastcall AddUser();
```

Description

Call AddUser to add a user to the user table.

Note: You must install InterBase 6 to use this feature.

12.1.13.4.2 TIBSecurityService.DeleteUser Method

Deletes a user from the user table.

Pascal

```
procedure DeleteUser;
```

C++

```
__fastcall DeleteUser();
```

Description

Call DeleteUser to delete a user from the user table.

Note: You must install InterBase 6 to use this feature.

12.1.13.4.3 TIBSecurityService.DisplayUser Method

Fetches user information.

Pascal

```
procedure DisplayUser(UserName: string);
```

C++

```
__fastcall DisplayUser(UnicodeString UserName);
```

Description

Call DisplayUser to fetch user information and dump it into the TUserInfo (see page 472) record, which can then be returned by the UserInfo (see page 451) property.

Note: You must install InterBase 6 to use this feature.

12.1.13.4.4 TIBSecurityService.DisplayUsers Method

Fetches information for all users.

Pascal

```
procedure DisplayUsers;
```

C++

```
__fastcall DisplayUsers();
```

Description

Call DisplayUsers to fetch all valid user information, which can then be returned by the UserInfo (see page 451) property.

Note: You must install InterBase 6 to use this feature.

12.1.13.4.5 TIBSecurityService.Loaded Method

Initializes the component after the form file has been read into memory.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

Do not call the protected Loaded method. The streaming system calls this method after it loads the component's form from a stream.

When the streaming system loads a form or data module from its form file, it first constructs the form component by calling its constructor, then reads its property values from the form file. After reading all the property values for all the components, the streaming system calls the Loaded methods of each component in the order the components were created. This gives the components a chance to initialize any data that depends on the values of other components or other parts of itself.

Note: All references to sibling components are resolved by the time Loaded is called. Loaded is the first place that sibling pointers can be used after being streamed in.

As implemented in TComponent, Loaded clears the csLoading flag in the ComponentState property, indicating that the component is no longer loading.

Warning: Loaded may be called multiple times on inherited forms. It is called every time a level of inheritance is streamed in. Do not allocate memory in an overridden Loaded method without first checking that the memory has not been allocated in a previous call.

See Also

ComponentState

Create

TReader

12.1.13.4.6 TIBSecurityService.ModifyUser Method

Modifies user information.

Pascal

```
procedure ModifyUser;
```

C++

```
__fastcall ModifyUser();
```

Description

Call ModifyUser to modify a user's information.

Note: You must install InterBase 6 to use this feature.

12.1.13.4.7 TIBSecurityService.SetServiceStartOptions Method

Pascal

```
procedure SetServiceStartOptions; override;
```

C++

```
virtual __fastcall SetServiceStartOptions();
```

Description

This is SetServiceStartOptions, a member of class TIBSecurityService.

12.1.13.5 TIBSecurityService Properties

The properties of the TIBSecurityService class are listed here.

Public Properties

	Name	Description
R	UserInfo (see page 451)	Provides information about a user.
R	UserInfoCount (see page 451)	Returns the number of elements in the TUserInfo (see page 472) array.

Published Properties

	Name	Description
	FirstName (see page 449)	Sets or returns the user's first name.
	GroupID (see page 449)	Sets or returns the user's group ID number.
	LastName (see page 449)	Sets or returns the user's last name.
	MiddleName (see page 449)	Sets or returns the user's middle name.
	Password (see page 450)	Sets the user's password.
	SecurityAction (see page 450)	Specifies the security action.
	SQlRole (see page 450)	The role name used to connect to the isc4.gdb security database.
	UserID (see page 450)	Sets or returns the user's ID number.
	UserName (see page 451)	Sets or returns the user's login name.

12.1.13.5.1 TIBSecurityService.FirstName Property

Sets or returns the user's first name.

Pascal

```
property FirstName: string;
```

C++

```
__property UnicodeString FirstName;
```

Description

Use FirstName to set or return the user's first name.

Note: You must install InterBase 6 to use this feature.

12.1.13.5.2 TIBSecurityService.GroupID Property

Sets or returns the user's group ID number.

Pascal

```
property GroupID: Integer;
```

C++

```
__property int GroupID;
```

Description

Use GroupID to set or return the user's group identification number (GID).

Note: You must install InterBase 6 to use this feature.

12.1.13.5.3 TIBSecurityService.LastName Property

Sets or returns the user's last name.

Pascal

```
property LastName: string;
```

C++

```
__property UnicodeString LastName;
```

Description

Use LastName to set or return the user's last name.

Note: You must install InterBase 6 to use this feature.

12.1.13.5.4 TIBSecurityService.MiddleName Property

Sets or returns the user's middle name.

Pascal

```
property MiddleName: string;
```

C++

```
__property UnicodeString MiddleName;
```

Description

Use MiddleName to set or return user's middle name.

Note: You must install InterBase 6 to use this feature.

12.1.13.5.5 TIBSecurityService.Password Property

Sets the user's password.

Pascal

```
property Password: string;
```

C++

```
__property UnicodeString Password;
```

Description

Use Password to indicate the user's password when adding or modifying a user's information.

Note: You must install InterBase 6 to use this feature.

12.1.13.5.6 TIBSecurityService.SecurityAction Property

Specifies the security action.

Pascal

```
property SecurityAction: TSecurityAction;
```

C++

```
__property TSecurityAction SecurityAction;
```

Description

Use SecurityAction to indicate what operation the security server should perform.

Note: You must install InterBase 6 to use this feature.

12.1.13.5.7 TIBSecurityService.SQlRole Property

The role name used to connect to the isc4.gdb security database.

Pascal

```
property SQlRole: string;
```

C++

```
__property UnicodeString SQlRole;
```

Description

Use SQlRole to set or return role name used to connect to the isc4.gdb security database. For more information about using SQL roles with InterBase, refer to "ANSI SQL 3 roles" in the InterBase 6 Operations Guide

You must install InterBase 6 to use this feature.

12.1.13.5.8 TIBSecurityService.UserID Property

Sets or returns the user's ID number.

Pascal

```
property UserID: Integer;
```

C++

```
__property int UserID;
```

Description

Use UserID to set or return the user's identification number (UID).

Note: You must install InterBase 6 to use this feature.

12.1.13.5.9 TIBSecurityService.UserInfo Property

Provides information about a user.

Pascal

```
property UserInfo [Index: Integer]: TUserInfo;
```

C++

```
__property TUserInfo UserInfo[int Index];
```

Description

Use UserInfo to return the user's information as a TUserInfo (see page 472) record, which contains the user name, groupID, UserID (see page 450), and the user's first, middle, and last names.

Note: You must install InterBase 6 to use this feature.

12.1.13.5.10 TIBSecurityService.UserInfoCount Property

Returns the number of elements in the TUserInfo (see page 472) array.

Pascal

```
property UserInfoCount: Integer;
```

C++

```
__property int UserInfoCount;
```

Description

Use UserInfoCount to return the number of elements in the TUserInfo (see page 472) array.

Note: You must install InterBase 6 to use this feature.

12.1.13.5.11 TIBSecurityService.UserName Property

Sets or returns the user's login name.

Pascal

```
property UserName: string;
```

C++

```
__property UnicodeString UserName;
```

Description

Use UserName to set or return the user's login (user) name.

Note: You must install InterBase 6 to use this feature.

12.1.14 TIBServerProperties Class

TIBServerProperties returns database server information.

Class Hierarchy

Pascal

```
TIBServerProperties = class(TIBCustomService);
```

C++

```
class TIBServerProperties : public TIBCustomService;
```

File

IBServices

Description

TIBServerProperties returns database server information, including configuration parameters, and version and license information.

Note: You must install InterBase 6 to use this feature.

See Also

Getting Started with InterBase Express

12.1.14.1 TIBServerProperties Members

The following tables list the members exposed by TIBServerProperties.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.
V	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.

Protected Properties

	Name	Description
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.

Public Methods

	Name	Description
	Attach (see page 427)	Attaches to the database.
V	Create (see page 453)	Creates an instance of TIBServerProperties (see page 452)
V	Destroy (see page 453)	Destroys the instance of TIBServerProperties (see page 452).

	Detach (see page 428)	Detaches from the database.
	Fetch (see page 454)	Fetches information from the server.
	FetchConfigParams (see page 454)	Returns the database configuration parameters.
	FetchDatabaseInfo (see page 455)	Returns database information.
	FetchLicenseInfo (see page 455)	Returns license information.
	FetchLicenseMaskInfo (see page 455)	Returns license mask information.
	FetchVersionInfo (see page 455)	Returns the database version information.

Public Properties

	Name	Description
R	ConfigParams (see page 456)	Returns the server configuration parameters.
R	DatabaseInfo (see page 456)	Indicates database information.
R	Handle (see page 430)	Returns the database handle.
R	LicenseInfo (see page 457)	Indicates license information.
R	LicenseMaskInfo (see page 457)	Returns license mask information.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.
R	VersionInfo (see page 457)	Indicates version information.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Options (see page 457)	Indicates the server properties.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.

12.1.14.2 TIBServerProperties.Create Constructor

Creates an instance of TIBServerProperties (see page 452)

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBServerProperties(TComponent * AOwner);
```

Description

Call Create to create an instance of TIBServerProperties (see page 452).

Note: You must install InterBase 6 to use this feature.

12.1.14.3 TIBServerProperties.Destroy Destructor

Destroys the instance of TIBServerProperties (see page 452).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBServerProperties();
```

Description

Do not call Destroy directly in an application. Instead, an application should call Free. Free verifies that the service object has not already been freed before it calls Destroy.

Note: You must install InterBase 6 to use this feature.

12.1.14.4 TIBServerProperties Methods

The methods of the TIBServerProperties class are listed here.

Public Methods

	Name	Description
	Fetch (see page 454)	Fetches information from the server.
	FetchConfigParams (see page 454)	Returns the database configuration parameters.
	FetchDatabaseInfo (see page 455)	Returns database information.
	FetchLicenseInfo (see page 455)	Returns license information.
	FetchLicenseMaskInfo (see page 455)	Returns license mask information.
	FetchVersionInfo (see page 455)	Returns the database version information.

12.1.14.4.1 TIBServerProperties.Fetch Method

Fetches information from the server.

Pascal

```
procedure Fetch;
```

C++

```
__fastcall Fetch();
```

Description

Call Fetch to get information from the server.

Note: You must install InterBase 6 to use this feature.

12.1.14.4.2 TIBServerProperties.FetchConfigParams Method

Returns the database configuration parameters.

Pascal

```
procedure FetchConfigParams;
```

C++

```
__fastcall FetchConfigParams();
```

Description

Call FetchConfigParams to return database configuration parameters.

Note: You must install InterBase 6 to use this feature.

12.1.14.4.3 TIBServerProperties.FetchDatabaseInfo Method

Returns database information.

Pascal

```
procedure FetchDatabaseInfo;
```

C++

```
__fastcall FetchDatabaseInfo();
```

Description

Call FetchDatabaseInfo to return database information.

Note: You must install InterBase 6 to use this feature.

12.1.14.4.4 TIBServerProperties.FetchLicenseInfo Method

Returns license information.

Pascal

```
procedure FetchLicenseInfo;
```

C++

```
__fastcall FetchLicenseInfo();
```

Description

Call FetchLicenseInfo to return database license information.

Note: You must install InterBase 6 to use this feature.

12.1.14.4.5 TIBServerProperties.FetchLicenseMaskInfo Method

Returns license mask information.

Pascal

```
procedure FetchLicenseMaskInfo;
```

C++

```
__fastcall FetchLicenseMaskInfo();
```

Description

Call FetchLicenseMaskInfo to return database license mask information.

Note: You must install InterBase 6 to use this feature.

12.1.14.4.6 TIBServerProperties.FetchVersionInfo Method

Returns the database version information.

Pascal

```
procedure FetchVersionInfo;
```

C++

```
__fastcall FetchVersionInfo();
```

Description

Call FetchVersionInfo to return the database version information.

Note: You must install InterBase 6 to use this feature.

12.1.14.5 TIBServerProperties Properties

The properties of the TIBServerProperties class are listed here.

Public Properties

	Name	Description
	ConfigParams (see page 456)	Returns the server configuration parameters.
	DatabaseInfo (see page 456)	Indicates database information.
	LicenseInfo (see page 457)	Indicates license information.
	LicenseMaskInfo (see page 457)	Returns license mask information.
	VersionInfo (see page 457)	Indicates version information.

Published Properties

	Name	Description
	Options (see page 457)	Indicates the server properties.

12.1.14.5.1 TIBServerProperties.ConfigParams Property

Returns the server configuration parameters.

Pascal

```
property ConfigParams: TConfigParams;
```

C++

```
__property TConfigParams ConfigParams;
```

Description

Use the ConfigParams property to return the server configuration parameters as a TConfigParams (see page 401) record, which includes the configuration file data, base location, lock file location, message file location, and the security database location.

Note: You must install InterBase 6 to use this feature.

12.1.14.5.2 TIBServerProperties.DatabaseInfo Property

Indicates database information.

Pascal

```
property DatabaseInfo: TDatabaseInfo;
```

C++

```
__property TDatabaseInfo DatabaseInfo;
```

Description

Use the DatabaseInfo property to obtain database information, which includes the database name, number of attachments, and number of databases.

Note: You must install InterBase 6 to use this feature.

12.1.14.5.3 TIBServerProperties.LicenseInfo Property

Indicates license information.

Pascal

```
property LicenseInfo: TLicenseInfo;
```

C++

```
__property TLicenseInfo LicenseInfo;
```

Description

Use the LicenseInfo property to return licensing information as a TLicenseInfo (see page 466) record, which includes the key, ID, and number of licensed users.

Note: You must install InterBase 6 to use this feature.

12.1.14.5.4 TIBServerProperties.LicenseMaskInfo Property

Returns license mask information.

Pascal

```
property LicenseMaskInfo: TLicenseMaskInfo;
```

C++

```
__property TLicenseMaskInfo LicenseMaskInfo;
```

Description

Use the LicenseMaskInfo property to return licensing information as a TLicenseMaskInfo (see page 468) record, which includes the license mask and capability mask. A license mask is a bitmask representing the software activation certificate options currently enabled on the server. A capability mask is a bitmask representing the capabilities currently enabled on the server.

Note: You must install InterBase 6 to use this feature.

12.1.14.5.5 TIBServerProperties.Options Property

Indicates the server properties.

Pascal

```
property Options: TPropertyOptions;
```

C++

```
__property TPropertyOptions Options;
```

Description

Use the Options property to set and return server properties.

Note: You must install InterBase 6 to use this feature.

12.1.14.5.6 TIBServerProperties.VersionInfo Property

Indicates version information.

Pascal

```
property VersionInfo: TVersionInfo;
```

C++

```
__property TVersionInfo VersionInfo;
```

Description

Use the VersionInfo property to obtain the server configuration parameters as a TVersionInfo (see page 475) type, which includes the service version, and server version and implementation.

Note: You must install InterBase 6 to use this feature.

12.1.15 TIBStatisticalService Class

TIBStatisticalService allows you to view database statistics.

Class Hierarchy

Pascal

```
TIBStatisticalService = class(TIBControlAndQueryService);
```

C++

```
class TIBStatisticalService : public TIBControlAndQueryService;
```

File

IBServices

Description

Use a TIBStatisticalService object to view database statistics.

Note: You must install InterBase 6 to use this feature.

See Also

Getting Started with InterBase Express

12.1.15.1 TIBStatisticalService Members

The following tables list the members exposed by TIBStatisticalService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.
	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
V	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.
	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 460)	This is SetServiceStartOptions, a member of class TIBStatisticalService.

Protected Properties

	Name	Description
	Action (see page 420)	This is Action, a member of class TIBControlAndQueryService.
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Methods

	Name	Description
	Attach (see page 427)	Attaches to the database.
V	Create (see page 419)	Creates an instance of a TIBControlAndQueryService (see page 417).
V	Create (see page 422)	Creates an instance of TIBControlService (see page 421)
V	Create (see page 426)	Creates an instance of TIBCustomService (see page 425)
V	Destroy (see page 426)	Destroys the service.
	Detach (see page 428)	Detaches from the database.
	GetNextChunk (see page 419)	Gets the next chunk of data.
	GetNextLine (see page 420)	Starts the service.
V	ServiceStart (see page 423)	Starts the service.

Public Properties

	Name	Description
R	Eof (see page 421)	Indicates that the end of file has been reached.
R	Handle (see page 430)	Returns the database handle.
R	IsServiceRunning (see page 424)	Indicates whether the service is running.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	BufferSize (see page 420)	Sets or returns the buffer size.
	DatabaseName (see page 460)	Sets or returns the database name.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Options (see page 460)	Indicates the desired statistical information.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.

12.1.15.2 TIBStatisticalService Methods

The methods of the TIBStatisticalService class are listed here.

Protected Methods

	Name	Description
V	SetServiceStartOptions (see page 460)	This is SetServiceStartOptions, a member of class TIBStatisticalService.

12.1.15.2.1 TIBStatisticalService.SetServiceStartOptions Method

Pascal

```
procedure SetServiceStartOptions; override;
```

C++

```
virtual __fastcall SetServiceStartOptions();
```

Description

This is SetServiceStartOptions, a member of class TIBStatisticalService.

12.1.15.3 TIBStatisticalService Properties

The properties of the TIBStatisticalService class are listed here.

Published Properties

	Name	Description
	DatabaseName (see page 460)	Sets or returns the database name.
	Options (see page 460)	Indicates the desired statistical information.

12.1.15.3.1 TIBStatisticalService.DatabaseName Property

Sets or returns the database name.

Pascal

```
property DatabaseName: string;
```

C++

```
__property UnicodeString DatabaseName;
```

Description

Use DatabaseName to set or return the database name.

Note: You must install InterBase 6 to use this feature.

12.1.15.3.2 TIBStatisticalService.Options Property

Indicates the desired statistical information.

Pascal

```
property Options: TStatOptions;
```

C++

```
__property TStatOptions Options;
```

Description

Use the Options property to request database statistics. These options are incremental; that is, including DbLog also returns HeaderPages statistics, including IndexPages also returns DbLog and HeaderPages statistics, and so forth.

Note: You must install InterBase 6 to use this feature.

12.1.16 TIBValidationService Class

TIBValidationService enables you to validate your database and reconcile your database transactions.

Class Hierarchy

Pascal

```
TIBValidationService = class(TIBControlAndQueryService);
```

C++

```
class TIBValidationService : public TIBControlAndQueryService;
```

File

IBServices

Description

Use a TIBValidationService object to validate your database and reconcile your database transactions.

Note: You must install InterBase 6 to use this feature.

See Also

Getting Started with InterBase Express

12.1.16.1 TIBValidationService Members

The following tables list the members exposed by TIBValidationService.

Protected Methods

	Name	Description
	CheckActive (see page 427)	This is CheckActive, a member of class TIBCustomService.
	CheckInactive (see page 428)	This is CheckInactive, a member of class TIBCustomService.
	InternalServiceQuery (see page 428)	This is InternalServiceQuery, a member of class TIBCustomService.
	InternalServiceStart (see page 423)	This is InternalServiceStart, a member of class TIBControlService.
V	Loaded (see page 428)	Initializes the component after the form file has been read into memory.
	Login (see page 429)	This is Login, a member of class TIBCustomService.
	ServiceStartAddParam (see page 423)	This is the overview for the ServiceStartAddParam method overload.
V	SetServiceStartOptions (see page 464)	This is SetServiceStartOptions, a member of class TIBValidationService.

Protected Properties

	Name	Description
	Action (see page 420)	This is Action, a member of class TIBControlAndQueryService.
	BufferSize (see page 430)	This is BufferSize, a member of class TIBCustomService.
R	OutputBuffer (see page 431)	This is OutputBuffer, a member of class TIBCustomService.
	OutputBufferOption (see page 431)	This is OutputBufferOption, a member of class TIBCustomService.
	ServiceQueryParams (see page 432)	This is ServiceQueryParams, a member of class TIBCustomService.
	ServiceStartParams (see page 425)	This is ServiceStartParams, a member of class TIBControlService.

Public Methods

	Name	Description
	Attach (see page 427)	Attaches to the database.

	Create (see page 462)	Creates an instance of TIBValidationService (see page 461)
	Destroy (see page 463)	Destroys the service.
	Detach (see page 428)	Detaches from the database.
	FetchLimboTransactionInfo (see page 463)	Gets limbo transaction information.
	FixLimboTransactionErrors (see page 463)	Fixes limbo transaction errors.
	GetNextChunk (see page 419)	Gets the next chunk of data.
	GetNextLine (see page 420)	Starts the service.
	ServiceStart (see page 423)	Starts the service.

Public Properties

	Name	Description
	Eof (see page 421)	Indicates that the end of file has been reached.
	Handle (see page 430)	Returns the database handle.
	IsServiceRunning (see page 424)	Indicates whether the service is running.
	LimboTransactionInfo (see page 465)	Returns the limbo transaction information.
	LimboTransactionInfoCount (see page 465)	Returns the number of elements in the TLimboTransactionInfo (see page 469) array.
	ServiceParamBySPB (see page 432)	Returns and sets SPB parameters.

Published Events

	Name	Description
	OnAttach (see page 433)	Occurs when the database is attached.
	OnLogin (see page 433)	Occurs during database login.

Published Properties

	Name	Description
	Active (see page 430)	Sets the active state of the service.
	BufferSize (see page 420)	Sets or returns the buffer size.
	DatabaseName (see page 464)	Sets or returns the database name.
	GlobalAction (see page 465)	Indicates the global transaction action.
	LoginPrompt (see page 430)	Turns on and off login prompt display.
	Options (see page 465)	Indicates the options for the database restore.
	Params (see page 431)	Sets or returns the database parameters
	Protocol (see page 431)	Sets or returns the network protocol.
	ServerName (see page 431)	Sets the name of the server on which the services are to be run.
	TraceFlags (see page 432)	This is TraceFlags, a member of class TIBCustomService.

12.1.16.2 TIBValidationService.Create Constructor

Creates an instance of TIBValidationService (see page 461)

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBValidationService(TComponent * AOwner);
```

Description

Call Create to create an instance of TIBValidationService (see page 461).

Note: You must install InterBase 6 to use this feature.

12.1.16.3 TIBValidationService.Destroy Destructor

Destroys the service.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBValidationService();
```

Description

Do not call Destroy directly in an application. Instead, an application should call Free. Free verifies that the service object has not already been freed before it calls Destroy.

Note: You must install InterBase 6 to use this feature.

12.1.16.4 TIBValidationService Methods

The methods of the TIBValidationService class are listed here.

Protected Methods

	Name	Description
	SetServiceStartOptions (see page 464)	This is SetServiceStartOptions, a member of class TIBValidationService.

Public Methods

	Name	Description
	FetchLimboTransactionInfo (see page 463)	Gets limbo transaction information.
	FixLimboTransactionErrors (see page 463)	Fixes limbo transaction errors.

12.1.16.4.1 TIBValidationService.FetchLimboTransactionInfo Method

Gets limbo transaction information.

Pascal

```
procedure FetchLimboTransactionInfo;
```

C++

```
__fastcall FetchLimboTransactionInfo();
```

Description

Call FetchLimboTransactionInfo to get limbo transaction information from the LimboTransactionInfo (see page 465) property.

Note: You must install InterBase 6 to use this feature.

12.1.16.4.2 TIBValidationService.FixLimboTransactionErrors Method

Fixes limbo transaction errors.

Pascal

```
procedure FixLimboTransactionErrors;
```

C++

```
__fastcall FixLimboTransactionErrors();
```

Description

Call FixLimboTransactionErrors to repair limbo transaction errors.

Note: You must install InterBase 6 to use this feature.

12.1.16.4.3 TIBValidationService.SetServiceStartOptions Method

Pascal

```
procedure SetServiceStartOptions; override;
```

C++

```
virtual __fastcall SetServiceStartOptions();
```

Description

This is SetServiceStartOptions, a member of class TIBValidationService.

12.1.16.5 TIBValidationService Properties

The properties of the TIBValidationService class are listed here.

Public Properties

	Name	Description
	LimboTransactionInfo (see page 465)	Returns the limbo transaction information.
	LimboTransactionInfoCount (see page 465)	Returns the number of elements in the TLimboTransactionInfo (see page 469) array.

Published Properties

	Name	Description
	DatabaseName (see page 464)	Sets or returns the database name.
	GlobalAction (see page 465)	Indicates the global transaction action.
	Options (see page 465)	Indicates the options for the database restore.

12.1.16.5.1 TIBValidationService.DatabaseName Property

Sets or returns the database name.

Pascal

```
property DatabaseName: string;
```

C++

```
__property UnicodeString DatabaseName;
```

Description

Use DatabaseName to set or return the name of the database to be repaired.

Note: You must install InterBase 6 to use this feature.

12.1.16.5.2 TIBValidationService.GlobalAction Property

Indicates the global transaction action.

Pascal

```
property GlobalAction: TTransactionGlobalAction;
```

C++

```
__property TTransactionGlobalAction GlobalAction;
```

Description

Use GlobalAction to set or return the global transaction action. This value indicates what action to take with the transactions that follow.

Note: You must install InterBase 6 to use this feature.

12.1.16.5.3 TIBValidationService.LimboTransactionInfo Property

Returns the limbo transaction information.

Pascal

```
property LimboTransactionInfo [Index: integer]: TLimboTransactionInfo;
```

C++

```
__property TLimboTransactionInfo LimboTransactionInfo[int Index];
```

Description

Use LimboTransactionInfo to return the limbo transaction information.

Note: You must install InterBase 6 to use this feature.

12.1.16.5.4 TIBValidationService.LimboTransactionInfoCount Property

Returns the number of elements in the TLimboTransactionInfo (see page 469) array.

Pascal

```
property LimboTransactionInfoCount: Integer;
```

C++

```
__property int LimboTransactionInfoCount;
```

Description

Use LimboTransactionInfoCount to set or return the number of elements in the TLimboTransactionInfo (see page 469) array.

Note: You must install InterBase 6 to use this feature.

12.1.16.5.5 TIBValidationService.Options Property

Indicates the options for the database restore.

Pascal

```
property Options: TValidateOptions;
```

C++

```
__property TValidateOptions Options;
```

Description

Use Options to select the options for the database restore.

Note: You must install InterBase 6 to use this feature.

12.1.17 TLicenseInfo Class

TLicenseInfo stores information about licensed users.

Class Hierarchy

Pascal

```
TLicenseInfo = class;
```

C++

```
class TLicenseInfo;
```

File

IBServices

Description

Key (see page 468) lists the license key for each licensed user.

Id (see page 467) lists the user Id (see page 467) for each licensed user.

Desc (see page 467) lists the description of each licensed user.

LicensedUsers (see page 468) indicates the number of users (the number of entries in the lists).

12.1.17.1 TLicenseInfo Members

The following tables list the members exposed by TLicenseInfo.

Public Fields

	Name	Description
	Desc (see page 467)	This is Desc, a member of class TLicenseInfo.
	Id (see page 467)	This is Id, a member of class TLicenseInfo.
	Key (see page 468)	This is Key, a member of class TLicenseInfo.
	LicensedUsers (see page 468)	This is LicensedUsers, a member of class TLicenseInfo.

Public Methods

	Name	Description
	Create (see page 467)	This is Create, a member of class TLicenseInfo.
	Destroy (see page 467)	This is Destroy, a member of class TLicenseInfo.

12.1.17.2 TLicenseInfo.Create Constructor

Pascal

```
constructor Create;
```

C++

```
__fastcall TLicenseInfo();
```

Description

This is Create, a member of class TLicenseInfo.

12.1.17.3 TLicenseInfo.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TLicenseInfo();
```

Description

This is Destroy, a member of class TLicenseInfo.

12.1.17.4 TLicenseInfo Fields

The fields of the TLicenseInfo class are listed here.

Public Fields

	Name	Description
	Desc (see page 467)	This is Desc, a member of class TLicenseInfo.
	Id (see page 467)	This is Id, a member of class TLicenseInfo.
	Key (see page 468)	This is Key, a member of class TLicenseInfo.
	LicensedUsers (see page 468)	This is LicensedUsers, a member of class TLicenseInfo.

12.1.17.4.1 TLicenseInfo.Desc Field

Pascal

```
Desc: array of string;
```

C++

```
array of string Desc;
```

Description

This is Desc, a member of class TLicenseInfo.

12.1.17.4.2 TLicenseInfo.Id Field

Pascal

```
Id: array of string;
```

C++

```
array of string Id;
```

Description

This is Id, a member of class TLicenseInfo.

12.1.17.4.3 TLicenseInfo.Key Field

Pascal

```
Key: array of string;
```

C++

```
array of string Key;
```

Description

This is Key, a member of class TLicenseInfo.

12.1.17.4.4 TLicenseInfo.LicensedUsers Field

Pascal

```
LicensedUsers: Integer;
```

C++

```
int LicensedUsers;
```

Description

This is LicensedUsers, a member of class TLicenseInfo.

12.1.18 TLicenseMaskInfo Class

TLicenseMaskInfo indicates the software activation certificate options enabled on the server.

Class Hierarchy

Pascal

```
TLicenseMaskInfo = class;
```

C++

```
class TLicenseMaskInfo;
```

File

IBServices

Description

LicenseMask (see page 469) is a bitmask representing the software activation certificate options currently enabled on the server.

CapabilityMask (see page 469) is a bitmask representing the capabilities currently enabled on the server.

12.1.18.1 TLicenseMaskInfo Members

The following tables list the members exposed by TLicenseMaskInfo.

Public Fields

	Name	Description
	CapabilityMask (see page 469)	This is CapabilityMask, a member of class TLicenseMaskInfo.
	LicenseMask (see page 469)	This is LicenseMask, a member of class TLicenseMaskInfo.

12.1.18.2 TLicenseMaskInfo Fields

The fields of the TLicenseMaskInfo class are listed here.

Public Fields

	Name	Description
	CapabilityMask (see page 469)	This is CapabilityMask, a member of class TLicenseMaskInfo.
	LicenseMask (see page 469)	This is LicenseMask, a member of class TLicenseMaskInfo.

12.1.18.2.1 TLicenseMaskInfo.CapabilityMask Field

Pascal

```
CapabilityMask: Integer;
```

C++

```
int CapabilityMask;
```

Description

This is CapabilityMask, a member of class TLicenseMaskInfo.

12.1.18.2.2 TLicenseMaskInfo.LicenseMask Field

Pascal

```
LicenseMask: Integer;
```

C++

```
int LicenseMask;
```

Description

This is LicenseMask, a member of class TLicenseMaskInfo.

12.1.19 TLimboTransactionInfo Class

TLimboTransactionInfo stores information about a limbo transaction.

Class Hierarchy

IBServices.TLimboTransactionInfo

Pascal

```
TLimboTransactionInfo = class;
```

C++

```
class TLimboTransactionInfo;
```

File

IBServices

Description

TLimboTransactionInfo describes limbo transaction. Limbo transactions are usually caused by the failure of a two-phased commit. They can also exist due to system failure or when a single-database transaction is prepared..

MultiDatabase (see page 472) indicates whether the limbo transaction involves multiple databases.

Transaction ID (see page 471) is the identifier for the limbo transaction.

Host site is the host name for the server where the transaction was started.

RemoteSite (see page 472) is the host name for a remote server involved in the transaction.

RemoteDatabasePath (see page 472) is the path to the remote server.

State (see page 472) is the current state of the limbo transaction.

Advise (see page 471) is the suggested action for ending the transaction.

Action (see page 471) is the action to take for ending the transaction.

12.1.19.1 TLimboTransactionInfo Members

The following tables list the members exposed by TLimboTransactionInfo.

Public Fields

	Name	Description
	Action (see page 471)	This is Action, a member of class TLimboTransactionInfo.
	Advise (see page 471)	This is Advise, a member of class TLimboTransactionInfo.
	HostSite (see page 471)	This is HostSite, a member of class TLimboTransactionInfo.
	ID (see page 471)	This is ID, a member of class TLimboTransactionInfo.
	MultiDatabase (see page 472)	This is MultiDatabase, a member of class TLimboTransactionInfo.
	RemoteDatabasePath (see page 472)	This is RemoteDatabasePath, a member of class TLimboTransactionInfo.
	RemoteSite (see page 472)	This is RemoteSite, a member of class TLimboTransactionInfo.
	State (see page 472)	This is State, a member of class TLimboTransactionInfo.

12.1.19.2 TLimboTransactionInfo Fields

The fields of the TLimboTransactionInfo class are listed here.

Public Fields

	Name	Description
	Action (see page 471)	This is Action, a member of class TLimboTransactionInfo.
	Advise (see page 471)	This is Advise, a member of class TLimboTransactionInfo.
	HostSite (see page 471)	This is HostSite, a member of class TLimboTransactionInfo.
	ID (see page 471)	This is ID, a member of class TLimboTransactionInfo.
	MultiDatabase (see page 472)	This is MultiDatabase, a member of class TLimboTransactionInfo.

	RemoteDatabasePath (see page 472)	This is RemoteDatabasePath, a member of class TLimboTransactionInfo.
	RemoteSite (see page 472)	This is RemoteSite, a member of class TLimboTransactionInfo.
	State (see page 472)	This is State, a member of class TLimboTransactionInfo.

12.1.19.2.1 TLimboTransactionInfo.Action Field

Pascal

```
Action: TTransactionAction;
```

C++

```
TTransactionAction Action;
```

Description

This is Action, a member of class TLimboTransactionInfo.

12.1.19.2.2 TLimboTransactionInfo.Advise Field

Pascal

```
Advise: TTransactionAdvise;
```

C++

```
TTransactionAdvise Advise;
```

Description

This is Advise, a member of class TLimboTransactionInfo.

12.1.19.2.3 TLimboTransactionInfo.HostSite Field

Pascal

```
HostSite: String;
```

C++

```
UnicodeString HostSite;
```

Description

This is HostSite, a member of class TLimboTransactionInfo.

12.1.19.2.4 TLimboTransactionInfo.ID Field

Pascal

```
ID: Integer;
```

C++

```
int ID;
```

Description

This is ID, a member of class TLimboTransactionInfo.

12.1.19.2.5 TLimboTransactionInfo.MultiDatabase Field

Pascal

```
MultiDatabase: Boolean;
```

C++

```
Boolean MultiDatabase;
```

Description

This is MultiDatabase, a member of class TLimboTransactionInfo.

12.1.19.2.6 TLimboTransactionInfo.RemoteDatabasePath Field

Pascal

```
RemoteDatabasePath: String;
```

C++

```
UnicodeString RemoteDatabasePath;
```

Description

This is RemoteDatabasePath, a member of class TLimboTransactionInfo.

12.1.19.2.7 TLimboTransactionInfo.RemoteSite Field

Pascal

```
RemoteSite: String;
```

C++

```
UnicodeString RemoteSite;
```

Description

This is RemoteSite, a member of class TLimboTransactionInfo.

12.1.19.2.8 TLimboTransactionInfo.State Field

Pascal

```
State: TTransactionState;
```

C++

```
TTransactionState State;
```

Description

This is State, a member of class TLimboTransactionInfo.

12.1.20 TUserInfo Class

TUserInfo stores information about an InterBase user for the security service.

Class Hierarchy

Pascal

```
TUserInfo = class;
```

C++

```
class TUserInfo;
```

File

IBServices

Description

TUserInfo is the user descriptor that the InterBase security service uses to describe a single user.

UserName (see page 475) is the user's login name.

FirstName (see page 474) is the user's first name.

MiddleName (see page 474) is the user's middle name.

LastName (see page 474) is the user's Last name.

GroupID (see page 474) is the user's group ID.

UserID (see page 474) is the user's individual ID.

12.1.20.1 TUserInfo Members

The following tables list the members exposed by TUserInfo.

Public Fields

	Name	Description
	FirstName (see page 474)	This is FirstName, a member of class TUserInfo.
	GroupID (see page 474)	This is GroupID, a member of class TUserInfo.
	LastName (see page 474)	This is LastName, a member of class TUserInfo.
	MiddleName (see page 474)	This is MiddleName, a member of class TUserInfo.
	UserID (see page 474)	This is UserID, a member of class TUserInfo.
	UserName (see page 475)	This is UserName, a member of class TUserInfo.

12.1.20.2 TUserInfo Fields

The fields of the TUserInfo class are listed here.

Public Fields

	Name	Description
	FirstName (see page 474)	This is FirstName, a member of class TUserInfo.
	GroupID (see page 474)	This is GroupID, a member of class TUserInfo.
	LastName (see page 474)	This is LastName, a member of class TUserInfo.
	MiddleName (see page 474)	This is MiddleName, a member of class TUserInfo.
	UserID (see page 474)	This is UserID, a member of class TUserInfo.
	UserName (see page 475)	This is UserName, a member of class TUserInfo.

12.1.20.2.1 TUserInfo.FirstName Field

Pascal

```
FirstName: string;
```

C++

```
UnicodeString FirstName;
```

Description

This is FirstName, a member of class TUserInfo.

12.1.20.2.2 TUserInfo.GroupID Field

Pascal

```
GroupID: Integer;
```

C++

```
int GroupID;
```

Description

This is GroupID, a member of class TUserInfo.

12.1.20.2.3 TUserInfo.LastName Field

Pascal

```
LastName: string;
```

C++

```
UnicodeString LastName;
```

Description

This is LastName, a member of class TUserInfo.

12.1.20.2.4 TUserInfo.MiddleName Field

Pascal

```
MiddleName: string;
```

C++

```
UnicodeString MiddleName;
```

Description

This is MiddleName, a member of class TUserInfo.

12.1.20.2.5 TUserInfo.UserID Field

Pascal

```
UserID: Integer;
```

C++

```
int UserID;
```

Description

This is UserID, a member of class TUserInfo.

12.1.20.2.6 TUserInfo.UserName Field

Pascal

```
UserName: string;
```

C++

```
UnicodeString UserName;
```

Description

This is UserName, a member of class TUserInfo.

12.1.21 TVersionInfo Class

TVersionInfo represents version information about an InterBase server.

Class Hierarchy

IBServices.TVersionInfo

Pascal

```
TVersionInfo = class;
```

C++

```
class TVersionInfo;
```

File

IBServices

Description

TVersionInfo supplies the following version information:

ServerVersion (see page 476) gives the version of the InterBase server.

ServerImplementation (see page 476) gives the implementation string.

ServiceVersion (see page 476) give the version number for the service that fetches the version information.

12.1.21.1 TVersionInfo Members

The following tables list the members exposed by TVersionInfo.

Public Fields

	Name	Description
	ServerImplementation (see page 476)	This is ServerImplementation, a member of class TVersionInfo.
	ServerVersion (see page 476)	This is ServerVersion, a member of class TVersionInfo.
	ServiceVersion (see page 476)	This is ServiceVersion, a member of class TVersionInfo.

12.1.21.2 TVersionInfo Fields

The fields of the TVersionInfo class are listed here.

Public Fields

	Name	Description
	ServerImplementation (see page 476)	This is ServerImplementation, a member of class TVersionInfo.
	ServerVersion (see page 476)	This is ServerVersion, a member of class TVersionInfo.
	ServiceVersion (see page 476)	This is ServiceVersion, a member of class TVersionInfo.

12.1.21.2.1 TVersionInfo.ServerImplementation Field

Pascal

```
ServerImplementation: string;
```

C++

```
UnicodeString ServerImplementation;
```

Description

This is ServerImplementation, a member of class TVersionInfo.

12.1.21.2.2 TVersionInfo.ServerVersion Field

Pascal

```
ServerVersion: String;
```

C++

```
UnicodeString ServerVersion;
```

Description

This is ServerVersion, a member of class TVersionInfo.

12.1.21.2.3 TVersionInfo.ServiceVersion Field

Pascal

```
ServiceVersion: Integer;
```

C++

```
int ServiceVersion;
```

Description

This is ServiceVersion, a member of class TVersionInfo.

12.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TBackupOption (see page 477)	This is record IBServices.TBackupOption.
	TLicensingAction (see page 478)	TLicensingAction indicates whether to add or remove InterBase software activation certificates.
	TOutputBufferOption (see page 478)	This is record IBServices.TOutputBufferOption.
	TPropertyOption (see page 479)	This is record IBServices.TPropertyOption.
	TProtocol (see page 479)	TProtocol identifies a network protocol.
	TRestoreOption (see page 480)	This is record IBServices.TRestoreOption.
	TSecurityAction (see page 480)	TSecurityAction indicates what operation the InterBase security service should perform.
	TSecurityModifyParam (see page 481)	This is record IBServices.TSecurityModifyParam.
	TShutdownMode (see page 481)	TShutdownMode indicates what steps to take when shutting down the InterBase database.
	TStatOption (see page 482)	This is record IBServices.TStatOption.
	TTransactionAction (see page 482)	TTransactionAction indicates the action a transaction can take when it times out.
	TTransactionAdvise (see page 483)	This is record IBServices.TTransactionAdvise.
	TTransactionGlobalAction (see page 483)	TTransactionGlobalAction indicates the action to take on a set of transactions.
	TTransactionState (see page 484)	This is record IBServices.TTransactionState.
	TValidateOption (see page 485)	This is record IBServices.TValidateOption.

12.2.1 IBServices.TBackupOption Enumeration

Pascal

```
TBackupOption = (
  IgnoreChecksums,
  IgnoreLimbo,
  MetadataOnly,
  NoGarbageCollection,
  OldMetadataDesc,
  NonTransportable,
  ConvertExtTables
);
```

C++

```
enum TBackupOption {
  IgnoreChecksums,
  IgnoreLimbo,
  MetadataOnly,
  NoGarbageCollection,
  OldMetadataDesc,
  NonTransportable,
  ConvertExtTables
};
```

File

IBServices

Description

This is record IBServices.TBackupOption.

12.2.2 IBServices.TLicensingAction Enumeration

TLicensingAction indicates whether to add or remove InterBase software activation certificates.

Pascal

```
TLicensingAction = (
  LicenseAdd,
  LicenseRemove
);
```

C++

```
enum TLicensingAction {
  LicenseAdd,
  LicenseRemove
};
```

File

IBServices

Description

TLicensingAction values are:

Value Meaning

LicenseAdd Add an InterBase software activation certificate.

LicenseRemove Remove an InterBase software activation certificate.

12.2.3 IBServices.TOutputBufferOption Enumeration

Pascal

```
TOutputBufferOption = (
  ByLine,
  ByChunk
);
```

C++

```
enum TOutputBufferOption {
  ByLine,
  ByChunk
};
```

File

IBServices

Description

This is record IBServices.TOutputBufferOption.

12.2.4 IBServices.TPropertyOption Enumeration

Pascal

```
TPropertyOption = (
  Database,
  License,
  LicenseMask,
  ConfigParameters,
  Version
);
```

C++

```
enum TPropertyOption {
  Database,
  License,
  LicenseMask,
  ConfigParameters,
  Version
};
```

File

IBServices

Description

This is record IBServices.TPropertyOption.

12.2.5 IBServices.TProtocol Enumeration

TProtocol identifies a network protocol.

Pascal

```
TProtocol = (
  TCP,
  SPX,
  NamedPipe,
  Local
);
```

C++

```
enum TProtocol {
  TCP,
  SPX,
  NamedPipe,
  Local
};
```

File

IBServices

Description

TProtocol values are:

Local	
NamedPipe	Named pipe (FIFO)
SPX	IPX/SPX
TCP	Transport control protocol

12.2.6 IBServices.TRestoreOption Enumeration

Pascal

```
TRestoreOption = (
  DeactivateIndexes,
  NoShadow,
  NoValidityCheck,
  OneRelationAtATime,
  Replace,
  CreateNewDB,
  UseAllSpace
);
```

C++

```
enum TRestoreOption {
  DeactivateIndexes,
  NoShadow,
  NoValidityCheck,
  OneRelationAtATime,
  Replace,
  CreateNewDB,
  UseAllSpace
};
```

File

IBServices

Description

This is record IBServices.TRestoreOption.

12.2.7 IBServices.TSecurityAction Enumeration

TSecurityAction indicates what operation the InterBase security service should perform.

Pascal

```
TSecurityAction = (
  ActionAddUser,
  ActionDeleteUser,
  ActionModifyUser,
  ActionDisplayUser
);
```

C++

```
enum TSecurityAction {
  ActionAddUser,
  ActionDeleteUser,
```

```
  ActionModifyUser,
  ActionDisplayUser
};
```

File

IBServices

Description

TSecurityAction values are:

ActionAddUser	Add the user
ActionDeleteUser	Delete the user
ActionDisplayUser	Display the user's information
ActionModifyUser	Modify the user account

12.2.8 IBServices.TSecurityModifyParam Enumeration

Pascal

```
TSecurityModifyParam = (
  ModifyFirstName,
  ModifyMiddleName,
  ModifyLastName,
  ModifyUserId,
  ModifyGroupId,
  ModifyPassword
);
```

C++

```
enum TSecurityModifyParam {
  ModifyFirstName,
  ModifyMiddleName,
  ModifyLastName,
  ModifyUserId,
  ModifyGroupId,
  ModifyPassword
};
```

File

IBServices

Description

This is record IBServices.TSecurityModifyParam.

12.2.9 IBServices.TShutdownMode Enumeration

TShutdownMode indicates what steps to take when shutting down the InterBase database.

Pascal

```
TShutdownMode = (
  Forced,
  DenyTransaction,
```

```
    DenyAttachment
  );
```

C++

```
  enum TShutdownMode {
    Forced,
    DenyTransaction,
    DenyAttachment
  };
```

File

IBServices

Description

TShutdownMode values are:

Forced	Shut down the database after the specified number of seconds. To shut down the database immediately, set the number of seconds to 0.
DenyTransaction	Deny new transactions for the specified number of seconds and then shut down. The shutdown will fail if there are still active transactions to the database.
DenyAttachment	Deny new attachments for the specified number of seconds and then shut down. The shutdown will fail if there are still active attachments to the database.

12.2.10 IBServices.TStatOption Enumeration

Pascal

```
  TStatOption = (
    DataPages,
    DbLog,
    HeaderPages,
    IndexPages,
    SystemRelations
  );
```

C++

```
  enum TStatOption {
    DataPages,
    DbLog,
    HeaderPages,
    IndexPages,
    SystemRelations
  };
```

File

IBServices

Description

This is record IBServices.TStatOption.

12.2.11 IBServices.TTransactionAction Enumeration

TTransactionAction indicates the action a transaction can take when it times out.

Pascal

```
TTransactionAction = (
  CommitAction,
  RollbackAction
);
```

C++

```
enum TTransactionAction {
  CommitAction,
  RollbackAction
};
```

File

IBServices

Description

TTransactionAction values are:

Value Meaning

TARollback Roll back the transaction.

TACommit Commit the transaction.

TARollbackRetaining Roll back the transaction, but retain the current transaction context. (Requires InterBase 6.0 or later)

TACommitRetaining Commit the transaction, but retain the current transaction context.

12.2.12 IBServices.TTransactionAdvise Enumeration

Pascal

```
TTransactionAdvise = (
  CommitAdvise,
  RollbackAdvise,
  UnknownAdvise
);
```

C++

```
enum TTransactionAdvise {
  CommitAdvise,
  RollbackAdvise,
  UnknownAdvise
};
```

File

IBServices

Description

This is record IBServices.TTransactionAdvise.

12.2.13 IBServices.TTransactionGlobalAction Enumeration

TTransactionGlobalAction indicates the action to take on a set of transactions.

Pascal

```
TTransactionGlobalAction = (
  CommitGlobal,
  RollbackGlobal,
  RecoverTwoPhaseGlobal,
  NoGlobalAction
);
```

C++

```
enum TTransactionGlobalAction {
  CommitGlobal,
  RollbackGlobal,
  RecoverTwoPhaseGlobal,
  NoGlobalAction
};
```

File

IBServices

Description

TTransactionGlobalAction values are:

Value Meaning

CommitGlobal Commit the transactions.

RollbackGlobal Roll back the transactions.

RecoverTwoPhaseGlobal Perform automated two-phase commit recovery on the transactions.

NoGlobalAction Take no action.

12.2.14 IBServices.TTransactionState Enumeration

Pascal

```
TTransactionState = (
  LimboState,
  CommitState,
  RollbackState,
  UnknownState
);
```

C++

```
enum TTransactionState {
  LimboState,
  CommitState,
  RollbackState,
  UnknownState
};
```

File

IBServices

Description

This is record IBServices.TTransactionState.

12.2.15 IBServices.TValidateOption Enumeration

Pascal

```
TValidateOption = (
  LimboTransactions,
  CheckDB,
  IgnoreChecksum,
  KillShadows,
  MendDB,
  SweepDB,
  ValidateDB,
  ValidateFull
);
```

C++

```
enum TValidateOption {
  LimboTransactions,
  CheckDB,
  IgnoreChecksum,
  KillShadows,
  MendDB,
  SweepDB,
  ValidateDB,
  ValidateFull
};
```

File

IBServices

Description

This is record IBServices.TValidateOption.

12.3 Types

The following table lists types in this documentation.

Types

Name	Description
TBackupOptions (see page 486)	TBackupOption (see page 477) and TBackupOptions indicate how TIBBackupService (see page 408) performs database backup.
TLoginEvent (see page 487)	This is type IBServices.TLoginEvent.
TPropertyOptions (see page 487)	TPropertyOption (see page 479) and TPropertyOptions indicate groups of InterBase server properties.
TRestoreOptions (see page 488)	TRestoreOption (see page 480) and TRestoreOptions indicate what operations take place when restoring an InterBase database.
TSecurityModifyParams (see page 489)	This is type IBServices.TSecurityModifyParams.
TStatOptions (see page 489)	TStatOption (see page 482) and TStatOptions indicate the different statistical information available about a database.
TValidateOptions (see page 490)	TValidateOption (see page 485) and TValidateOptions indicate what operations the validation service should perform when restoring a database.

12.3.1 IBServices.TBackupOptions Type

TBackupOption (see page 477) and TBackupOptions indicate how TIBBackupService (see page 408) performs database backup.

Pascal

```
TBackupOptions = set of TBackupOption;
```

C++

```
TBackupOption TBackupOptions;
```

File

IBServices

Description

TBackupOptions is a set of TBackupOption (see page 477) values. The following table lists the possible values:

IgnoreChecksums	A checksum is a page-by-page analysis of data to verify its integrity. A bad checksum means that a data page has been randomly overwritten, for example, due to a system crash. Checksum errors indicate data corruption, and InterBase normally prevents you from backing up a database if bad checksums are detected. Examine the data the next time you restore the database. To ignore checksums during backup, set IgnoreChecksums to true.
IgnoreLimbo	Limbo transactions are usually caused by the failure of a two-phased commit. They can also exist due to system failure or when a single-database transaction is prepared. To ignore limbo transactions during backup, set IgnoreLimbo to true. When your application ignores limbo transactions during backup, it ignores all records versions created by any limbo transaction, finds the most recently committed version of a record, and backs up that version.

MetadataOnly	When backing up a database, you can exclude its data, saving only the metadata. You might want to do this to: Retain a record of the metadata before it is modified. Create an empty copy of the database. The copy has the same metadata but can be populated with different data. Set MetaDataOnly to true to save only the metadata.
NoGarbageCollection	Garbage collections marks space used by old versions of data records as free for reuse. Generally, you want to perform garbage collection during backup. To disable garbage collection, set NoGarbageCollection to true.
OldMetadataDesc	Set OldMetadataDesc to true if you want to back up metadata in the old-style format.
NonTransportable	Set to NonTransportable to true if you do not want data to be moved to a machine with a different operating system than the one on which the backup was performed. If NonTransportable is set to false (the default), then data is saved in a generic format, enabling you to restore to any machine that supports InterBase.
ConvertExtTables	Set ConvertExtTables to true if you want to convert external files into internal tables during the backup

12.3.2 IBServices.TLoginEvent Type

Pascal

```
TLoginEvent = procedure (Database: TIBCustomService; LoginParams: TStrings) of object;
```

C++

```
(Database: TIBCustomService; LoginParams: TStrings) ( TLoginEvent)();
```

File

IBServices

Description

This is type IBServices.TLoginEvent.

12.3.3 IBServices.TPropertyOptions Type

TPropertyOption (see page 479) and TPropertyOptions indicate groups of InterBase server properties.

Pascal

```
TPropertyOptions = set of TPropertyOption;
```

C++

```
TPropertyOption TPropertyOptions;
```

File

IBServices

Description

TPropertyOptions is a set of TPropertyOption (see page 479) values. Each TPropertyOption (see page 479) value indicates a type of server property you wish to see. Each type of sever property has a corresponding data structure that holds the

information, as indicated in the following table:

Database	TDatabaseInfo (see page 404)
License	TLicenseInfo (see page 466)
ConfigParameters	TConfigParams (see page 401)
Version	TVersionInfo (see page 475)

12.3.4 IBServices.TRestoreOptions Type

TRestoreOption (see page 480) and TRestoreOptions indicate what operations take place when restoring an InterBase database.

Pascal

```
TRestoreOptions = set of TRestoreOption;
```

C++

```
TRestoreOption TRestoreOptions;
```

File

IBServices

Description

TRestoreOptions is a set of TRestoreOption (see page 480) values. The following table lists the possible values:

DeactivateIndexes	Normally, InterBase rebuilds indexes when it restores a database. If the database contained duplicate entries in a unique index when it was backed up, restoration fails. Duplicate values can be introduced into a database if indexes were temporarily made inactive (for example, to allow insertion of many records or to re-balance an index). To enable restoration to succeed in this case, set DeactivateIndexes to true. This makes indexes inactive and prevents them from rebuilding.
NoShadow	Normally, InterBase rebuilds previously defined database shadows when it restores a database. To make sure the shadows are not built, set the NoShadow option to true.
NoValidityCheck	If you redefine validity constraints in a database where data is already entered, your data might no longer satisfy the validity constraints. You might not discover this until you try to restore the database, at which time an error about invalid data appears. To restore a database that contains invalid data, set NoValidity to true. This deletes validity constraints from the metadata. After the database is restored, change the data to make it valid according to the new integrity constraints. Then add back the constraints that were deleted. **Important:** Always make a copy of metadata before redefining it; for example with the MetadataOnly option of the `Options` property of TIBBackupService (see page 408).
OneRelationAtATime	Set OneRelationATime to true to restore the metadata and data for each table together, one table at a time. This option is useful when you are having trouble restoring a backup file; for example, if the data is corrupt or invalid according to integrity constraints.
Replace	Your application cannot overwrite an existing database file unless you set Replace to true. If you attempt to restore to an existing database name and this option is not set, then the restore will not proceed. **Important:** Do not restore an existing database file when clients are operating on it. A safer approach is to rename the existing database file, restore the database, then drop or archive the old database as needed.

CreateNew	To restore a database to a new file, set the Create option to true.
UseAllSpace	When InterBase restores a database, it fills each data page to 80% of capacity by default. To set your application to restore a database with a 100% fill ratio on each data page, set the UseAllSpace option to true.

12.3.5 IBServices.TSecurityModifyParams Type

Pascal

```
TSecurityModifyParams = set of TSecurityModifyParam;
```

C++

```
TSecurityModifyParam TSecurityModifyParams;
```

File

IBServices

Description

This is type IBServices.TSecurityModifyParams.

12.3.6 IBServices.TStatOptions Type

TStatOption (see page 482) and TStatOptions indicate the different statistical information available about a database.

Pascal

```
TStatOptions = set of TStatOption;
```

C++

```
TStatOption TStatOptions;
```

File

IBServices

Description

TStatOptions is a set of TStatOption (see page 482) values. The following table lists the possible values:

Value Meaning

DataPages Request statistics for data tables in the database.

DbLog Stop reporting statistics after reporting information on the log pages.

HeaderPages Stop reporting statistics after reporting the information on the header page.

IndexPages Request statistics for the user indexes in the database.

SystemRelations Request statistics for system tables and indexes in addition to user tables and indexes.

12.3.7 IBServices.TValidateOptions Type

TValidateOption (see page 485) and TValidateOptions indicate what operations the validation service should perform when restoring a database.

Pascal

```
TValidateOptions = set of TValidateOption;
```

C++

```
TValidateOption TValidateOptions;
```

File

IBServices

Description

TValidateOptions is a set of TValidateOption (see page 485) values. The following table lists the possible values:

LimboTransactions	Return a list of limbo transactions
CheckDB	Validate the database, but don't repair it. Used in conjunction with ValidateDB.
IgnoreChecksum	Ignore all checksum errors
KillShadows	Kill all unavailable shadow files
MendDB	Prepare a corrupt database for backup
SweepDB	Perform a database sweep
ValidateDB	Validate the database structure
ValidateFull	Validate record fragments

12.4 Constants

The following table lists constants in this documentation.

Constants

Name	Description
DefaultBufferSize (see page 491)	This is constant IBServices.DefaultBufferSize.
SPBConstantNames (see page 491)	This is constant IBServices.SPBConstantNames.
SPBConstantValues (see page 492)	This is constant IBServices.SPBConstantValues.
SPBPrefix (see page 492)	This is constant IBServices.SPBPrefix.

12.4.1 IBServices.DefaultBufferSize Constant

Pascal

```
DefaultBufferSize = 32000;
```

C++

```
DefaultBufferSize = 32000;
```

File

IBServices

Description

This is constant IBServices.DefaultBufferSize.

12.4.2 IBServices.SPBConstantNames Constant

Pascal

```
SPBConstantNames: array[1..isc_spb_last_spb_constant] of String = ( 'user_name',
'sys_user_name', 'sys_user_name_enc', 'password', 'password_enc', 'command_line', 'db_name',
'verbose', 'options', 'connect_timeout', 'dummy_packet_interval', 'sql_role_name' );
```

C++

```
array[1..isc_spb_last_spb_constant] of String SPBConstantNames = ( 'user_name',
'sys_user_name', 'sys_user_name_enc', 'password', 'password_enc', 'command_line', 'db_name',
'verbose', 'options', 'connect_timeout', 'dummy_packet_interval', 'sql_role_name' );
```

File

IBServices

Description

This is constant IBServices.SPBConstantNames.

12.4.3 IBServices.SPBConstantValues Constant

Pascal

```
SPBConstantValues: array[1..isc_spb_last_spb_constant] of Integer = (
isc_spb_user_name_mapped_to_server, isc_spb_sys_user_name_mapped_to_server,
isc_spb_sys_user_name_enc_mapped_to_server, isc_spb_password_mapped_to_server,
isc_spb_password_enc_mapped_to_server, isc_spb_command_line_mapped_to_server,
isc_spb_dbname_mapped_to_server, isc_spb_verbose_mapped_to_server,
isc_spb_options_mapped_to_server, isc_spb_connect_timeout_mapped_to_server,
isc_spb_dummy_packet_interval_mapped_to_server, isc_spb_sql_role_name_mapped_to_server );
```

C++

```
array[1..isc_spb_last_spb_constant] of Integer SPBConstantValues = (
isc_spb_user_name_mapped_to_server, isc_spb_sys_user_name_mapped_to_server,
isc_spb_sys_user_name_enc_mapped_to_server, isc_spb_password_mapped_to_server,
isc_spb_password_enc_mapped_to_server, isc_spb_command_line_mapped_to_server,
isc_spb_dbname_mapped_to_server, isc_spb_verbose_mapped_to_server,
isc_spb_options_mapped_to_server, isc_spb_connect_timeout_mapped_to_server,
isc_spb_dummy_packet_interval_mapped_to_server, isc_spb_sql_role_name_mapped_to_server );
```

File

IBServices

Description

This is constant IBServices.SPBConstantValues.

12.4.4 IBServices.SPBPrefix Constant

Pascal

```
SPBPrefix = 'isc_spb_';
```

C++

```
SPBPrefix = 'isc_spb_';
```

File

IBServices

Description

This is constant IBServices.SPBPrefix.

13 IBSQL Namespace

This is namespace IBSQL.

Topics

Name	Description
Classes (see page 494)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 556)	The following table lists structs, records, enums in this documentation.
Types (see page 558)	The following table lists types in this documentation.

Classes

	Name	Description
	TIBBatch (see page 494)	TIBBatch is the abstract base class for the TIBBatchInput (see page 497) and TIBBatchOutput (see page 498) objects.
	TIBBatchInput (see page 497)	TIBBatchInput is the abstract class for performing all batch input.
	TIBBatchOutput (see page 498)	TIBBatchOutput is the abstract class for performing all batch output.
	TIBInputDelimitedFile (see page 500)	TIBInputDelimitedFile performs batch input of data from delimited files.
	TIBInputRawFile (see page 505)	TIBInputRawFile inputs data from a raw file.
	TIBOutputDelimitedFile (see page 507)	TIBOutputDelimitedFile performs batch output of data to delimited files.
	TIBOutputRawFile (see page 511)	TIBOutputRawFile outputs data to a raw file format.
	TIBSQL (see page 513)	TIBSQL provides an object for executing an InterBase SQL (see page 534) statement with minimal overhead.
	TIBXSQLDA (see page 536)	TIBXSQLDA provides properties and methods for use with the IBSQL component.
	TIBXSQLVAR (see page 544)	TIBXSQLVAR provides properties and methods for use with the IBSQL component.

Structs, Records, Enums

	Name	Description
	TIBSQLTypes (see page 556)	TIBSQLTypes indicates the type of an SQL query against an InterBase database.

Types

Name	Description
TIBXSQLVARArray (see page 558)	This is type IBSQL.TIBXSQLVARArray.

13.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBBatch (see page 494)	TIBBatch is the abstract base class for the TIBBatchInput (see page 497) and TIBBatchOutput (see page 498) objects.
	TIBBatchInput (see page 497)	TIBBatchInput is the abstract class for performing all batch input.
	TIBBatchOutput (see page 498)	TIBBatchOutput is the abstract class for performing all batch output.
	TIBInputDelimitedFile (see page 500)	TIBInputDelimitedFile performs batch input of data from delimited files.
	TIBInputRawFile (see page 505)	TIBInputRawFile inputs data from a raw file.
	TIBOutputDelimitedFile (see page 507)	TIBOutputDelimitedFile performs batch output of data to delimited files.
	TIBOutputRawFile (see page 511)	TIBOutputRawFile outputs data to a raw file format.
	TIBSQL (see page 513)	TIBSQL provides an object for executing an InterBase SQL (see page 534) statement with minimal overhead.
	TIBXSQLDA (see page 536)	TIBXSQLDA provides properties and methods for use with the IBSQL (see page 493) component.
	TIBXSQLVAR (see page 544)	TIBXSQLVAR provides properties and methods for use with the IBSQL (see page 493) component.

13.1.1 TIBBatch Class

TIBBatch is the abstract base class for the TIBBatchInput (see page 497) and TIBBatchOutput (see page 498) objects.

Class Hierarchy

Pascal

```
TIBBatch = class(TObject);
```

C++

```
class TIBBatch : public TObject;
```

File

IBSQL

Description

Use a TIBBatch object to provide properties and methods for use with the TIBBatchInput (see page 497) and TIBBatchOutput (see page 498) objects, which make it possible to input and output data in virtually any format.

Descendents of this class can specify a file name (for input or output), and a TIBXSQLDA (see page 536) object representing a record or parameters. The ReadyFile (see page 496) method is called right before performing the batch input or output.

13.1.1.1 TIBBatch Members

The following tables list the members exposed by TIBBatch.

Protected Fields

	Name	Description
	FColumns (see page 495)	This is FColumns, a member of class TIBBatch.
	FFilename (see page 495)	This is FFilename, a member of class TIBBatch.
	FParams (see page 496)	This is FParams, a member of class TIBBatch.

Public Methods

	Name	Description
	ReadyFile (see page 496)	Prepares the file.

Public Properties

	Name	Description
	Columns (see page 496)	Returns the XSQLDA columns.
	Filename (see page 497)	Sets or displays the name of the external file.
	Params (see page 497)	Returns the XSQLDA parameters.

13.1.1.2 TIBBatch Fields

The fields of the TIBBatch class are listed here.

Protected Fields

	Name	Description
	FColumns (see page 495)	This is FColumns, a member of class TIBBatch.
	FFilename (see page 495)	This is FFilename, a member of class TIBBatch.
	FParams (see page 496)	This is FParams, a member of class TIBBatch.

13.1.1.2.1 TIBBatch.FColumns Field

Pascal

```
FColumns: TIBXSQLDA;
```

C++

```
TIBXSQLDA FColumns;
```

Description

This is FColumns, a member of class TIBBatch.

13.1.1.2.2 TIBBatch.FFilename Field

Pascal

```
FFilename: String;
```

C++

```
UnicodeString FFilename;
```

Description

This is FFilename, a member of class TIBBatch.

13.1.1.2.3 TIBBatch.FParams Field

Pascal

```
FParams: TIBXSQLDA;
```

C++

```
TIBXSQLDA FParams;
```

Description

This is FParams, a member of class TIBBatch.

13.1.1.3 TIBBatch Methods

The methods of the TIBBatch class are listed here.

Public Methods

	Name	Description
	ReadyFile (see page 496)	Prepares the file.

13.1.1.3.1 TIBBatch.ReadyFile Method

Prepares the file.

Pascal

```
procedure ReadyFile; virtual; abstract;
```

C++

```
virtual __fastcall ReadyFile() = 0;
```

Description

Call ReadyFile to prepare the file for input or output.

13.1.1.4 TIBBatch Properties

The properties of the TIBBatch class are listed here.

Public Properties

	Name	Description
	Columns (see page 496)	Returns the XSQLDA columns.
	Filename (see page 497)	Sets or displays the name of the external file.
	Params (see page 497)	Returns the XSQLDA parameters.

13.1.1.4.1 TIBBatch.Columns Property

Returns the XSQLDA columns.

Pascal

```
property Columns: TIBXSQLDA;
```

C++

```
__property TIBXSQLDA Columns;
```

Description

Use the Columns property to retrieve or set the XSQLDA columns.

13.1.1.4.2 TIBBatch.Filename Property

Sets or displays the name of the external file.

Pascal

```
property Filename: String;
```

C++

```
__property UnicodeString Filename;
```

Description

Use the Filename property to set or display the external file name.

13.1.1.4.3 TIBBatch.Params Property

Returns the XSQLDA parameters.

Pascal

```
property Params: TIBXSQLDA;
```

C++

```
__property TIBXSQLDA Params;
```

Description

Use the Params property to retrieve or set the XSQLDA parameters.

13.1.2 TIBBatchInput Class

TIBBatchInput is the abstract class for performing all batch input.

Class Hierarchy

Pascal

```
TIBBatchInput = class(TIBBatch);
```

C++

```
class TIBBatchInput : public TIBBatch;
```

File

IBSQL

Description

Use a TIBBatchInput object to provide properties and methods for performing all batch input.

13.1.2.1 TIBBatchInput Members

The following tables list the members exposed by TIBBatchInput.

Protected Fields

	Name	Description
	FColumns (see page 495)	This is FColumns, a member of class TIBBatch.
	FFilename (see page 495)	This is FFilename, a member of class TIBBatch.
	FParams (see page 496)	This is FParams, a member of class TIBBatch.

Public Methods

	Name	Description
	ReadParameters (see page 498)	Reads the XSQLDA input parameters.
	ReadyFile (see page 496)	Prepares the file.

Public Properties

	Name	Description
	Columns (see page 496)	Returns the XSQLDA columns.
	Filename (see page 497)	Sets or displays the name of the external file.
	Params (see page 497)	Returns the XSQLDA parameters.

13.1.2.2 TIBBatchInput Methods

The methods of the TIBBatchInput class are listed here.

Public Methods

	Name	Description
	ReadParameters (see page 498)	Reads the XSQLDA input parameters.

13.1.2.2.1 TIBBatchInput.ReadParameters Method

Reads the XSQLDA input parameters.

Pascal

```
function ReadParameters: Boolean; virtual; abstract;
```

C++

```
virtual __fastcall Boolean ReadParameters() = 0;
```

Description

Call ReadParameters to read the input parameters of the extended SQL descriptor area (XSQLDA) from the specified file.

13.1.3 TIBBatchOutput Class

TIBBatchOutput is the abstract class for performing all batch output.

Class Hierarchy

Pascal

```
TIBBatchOutput = class(TIBBatch);
```

C++

```
class TIBBatchOutput : public TIBBatch;
```

File

IBSQL

Description

Use a TIBBatchOutput object to provide properties and methods for performing all batch output.

13.1.3.1 TIBBatchOutput Members

The following tables list the members exposed by TIBBatchOutput.

Protected Fields

	Name	Description
	FColumns (see page 495)	This is FColumns, a member of class TIBBatch.
	FFilename (see page 495)	This is FFilename, a member of class TIBBatch.
	FParams (see page 496)	This is FParams, a member of class TIBBatch.

Public Methods

	Name	Description
	ReadyFile (see page 496)	Prepares the file.
	WriteColumns (see page 499)	Outputs the data in columns in the XSQLDA to the specified file.

Public Properties

	Name	Description
	Columns (see page 496)	Returns the XSQLDA columns.
	Filename (see page 497)	Sets or displays the name of the external file.
	Params (see page 497)	Returns the XSQLDA parameters.

13.1.3.2 TIBBatchOutput Methods

The methods of the TIBBatchOutput class are listed here.

Public Methods

	Name	Description
	WriteColumns (see page 499)	Outputs the data in columns in the XSQLDA to the specified file.

13.1.3.2.1 TIBBatchOutput.WriteColumns Method

Outputs the data in columns in the XSQLDA to the specified file.

Pascal

```
function WriteColumns: Boolean; virtual; abstract;
```

C++

```
virtual __fastcall Boolean WriteColumns() = 0;
```

Description

Call WriteColumns to output data in columns in the extended SQL descriptor area (XSQLDA) to the specified file.

13.1.4 TIBInputDelimitedFile Class

TIBInputDelimitedFile performs batch input of data from delimited files.

Class Hierarchy

Pascal

```
TIBInputDelimitedFile = class(TIBBatchInput);
```

C++

```
class TIBInputDelimitedFile : public TIBBatchInput;
```

File

IBSQL

Description

Use a TIBInputDelimitedFile object to perform batch input of data from delimited files.

13.1.4.1 TIBInputDelimitedFile Members

The following tables list the members exposed by TIBInputDelimitedFile.

Protected Fields

	Name	Description
	FColDelimiter (see page 501)	This is FColDelimiter, a member of class TIBInputDelimitedFile.
	FColumns (see page 495)	This is FColumns, a member of class TIBBatch.
	FEOF (see page 501)	This is FEOF, a member of class TIBInputDelimitedFile.
	FFile (see page 502)	This is FFile, a member of class TIBInputDelimitedFile.
	FFilename (see page 495)	This is FFilename, a member of class TIBBatch.
	FLookAhead (see page 502)	This is FLookAhead, a member of class TIBInputDelimitedFile.
	FParams (see page 496)	This is FParams, a member of class TIBBatch.
	FReadBlanksAsNull (see page 502)	This is FReadBlanksAsNull, a member of class TIBInputDelimitedFile.
	FRowDelimiter (see page 502)	This is FRowDelimiter, a member of class TIBInputDelimitedFile.
	FSkipTitles (see page 503)	This is FSkipTitles, a member of class TIBInputDelimitedFile.

Public Methods

	Name	Description
	Destroy (see page 501)	Destroys the instance of TIBInputDelimitedFile (see page 500).
	GetColumn (see page 503)	Returns the contents of a column.
	ReadParameters (see page 503)	Reads the input parameters of the XSQLDA.
	ReadyFile (see page 503)	Prepares the output for the XSQLDA.

Public Properties

	Name	Description
	ColDelimiter (see page 504)	Sets the column delimiter for the input file.
	Columns (see page 496)	Returns the XSQLDA columns.

	Filename (see page 497)	Sets or displays the name of the external file.
	Params (see page 497)	Returns the XSQLDA parameters.
	ReadBlanksAsNull (see page 504)	Reads blank spaces in the input file as null characters.
	RowDelimiter (see page 504)	Sets the row delimiter for the input file.
	SkipTitles (see page 505)	Skips the first record of a delimited file.

13.1.4.2 TIBInputDelimitedFile.Destroy Destructor

Destroys the instance of TIBInputDelimitedFile (see page 500).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBInputDelimitedFile();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TIBInputDelimitedFile (see page 500) object is not already freed and only then calls Destroy.

13.1.4.3 TIBInputDelimitedFile Fields

The fields of the TIBInputDelimitedFile class are listed here.

Protected Fields

	Name	Description
	FColDelimiter (see page 501)	This is FColDelimiter, a member of class TIBInputDelimitedFile.
	FEOF (see page 501)	This is FEOF, a member of class TIBInputDelimitedFile.
	FFile (see page 502)	This is FFile, a member of class TIBInputDelimitedFile.
	FLookAhead (see page 502)	This is FLookAhead, a member of class TIBInputDelimitedFile.
	FReadBlanksAsNull (see page 502)	This is FReadBlanksAsNull, a member of class TIBInputDelimitedFile.
	FRowDelimiter (see page 502)	This is FRowDelimiter, a member of class TIBInputDelimitedFile.
	FSkipTitles (see page 503)	This is FSkipTitles, a member of class TIBInputDelimitedFile.

13.1.4.3.1 TIBInputDelimitedFile.FColDelimiter Field

Pascal

```
FColDelimiter: string;
```

C++

```
UnicodeString FColDelimiter;
```

Description

This is FColDelimiter, a member of class TIBInputDelimitedFile.

13.1.4.3.2 TIBInputDelimitedFile.FEOF Field

Pascal

```
FEOF: Boolean;
```

C++

```
Boolean FEOF;
```

Description

This is FEOF, a member of class TIBInputDelimitedFile.

13.1.4.3.3 TIBInputDelimitedFile.FFile Field

Pascal

```
FFile: TFileStream;
```

C++

```
TFileStream FFile;
```

Description

This is FFile, a member of class TIBInputDelimitedFile.

13.1.4.3.4 TIBInputDelimitedFile.FLookAhead Field

Pascal

```
FLookAhead: Char;
```

C++

```
Char FLookAhead;
```

Description

This is FLookAhead, a member of class TIBInputDelimitedFile.

13.1.4.3.5 TIBInputDelimitedFile.FReadBlanksAsNull Field

Pascal

```
FReadBlanksAsNull: Boolean;
```

C++

```
Boolean FReadBlanksAsNull;
```

Description

This is FReadBlanksAsNull, a member of class TIBInputDelimitedFile.

13.1.4.3.6 TIBInputDelimitedFile.FRowDelimiter Field

Pascal

```
FRowDelimiter: string;
```

C++

```
UnicodeString FRowDelimiter;
```

Description

This is FRowDelimiter, a member of class TIBInputDelimitedFile.

13.1.4.3.7 TIBInputDelimitedFile.FSkipTitles Field

Pascal

```
FSkipTitles: Boolean;
```

C++

```
Boolean FSkipTitles;
```

Description

This is FSkipTitles, a member of class TIBInputDelimitedFile.

13.1.4.4 TIBInputDelimitedFile Methods

The methods of the TIBInputDelimitedFile class are listed here.

Public Methods

	Name	Description
	GetColumn (see page 503)	Returns the contents of a column.
	ReadParameters (see page 503)	Reads the input parameters of the XSQLDA.
	ReadyFile (see page 503)	Prepares the output for the XSQLDA.

13.1.4.4.1 TIBInputDelimitedFile.GetColumn Method

Returns the contents of a column.

Pascal

```
function GetColumn(var Col: string): Integer;
```

C++

```
__fastcall int GetColumn(UnicodeString Col);
```

Description

Call GetColumn to return the contents of a column in the extended SQL descriptor area (XSQLDA).

13.1.4.4.2 TIBInputDelimitedFile.ReadParameters Method

Reads the input parameters of the XSQLDA.

Pascal

```
function ReadParameters: Boolean; override;
```

C++

```
virtual __fastcall Boolean ReadParameters();
```

Description

Call ReadParameters to read the input parameters of the extended SQL descriptor area (XSQLDA).

13.1.4.4.3 TIBInputDelimitedFile.ReadyFile Method

Prepares the output for the XSQLDA.

Pascal

```
procedure ReadyFile; override;
```

C++

```
virtual __fastcall ReadyFile();
```

Description

Call ReadyFile to prepare the output for the extended SQL descriptor area (XSQLDA).

13.1.4.5 TIBInputDelimitedFile Properties

The properties of the TIBInputDelimitedFile class are listed here.

Public Properties

	Name	Description
	ColDelimiter (see page 504)	Sets the column delimiter for the input file.
	ReadBlanksAsNull (see page 504)	Reads blank spaces in the input file as null characters.
	RowDelimiter (see page 504)	Sets the row delimiter for the input file.
	SkipTitles (see page 505)	Skips the first record of a delimited file.

13.1.4.5.1 TIBInputDelimitedFile.ColDelimiter Property

Sets the column delimiter for the input file.

Pascal

```
property ColDelimiter: string;
```

C++

```
__property UnicodeString ColDelimiter;
```

Description

Use ColDelimiter to set the column delimiter (either Tab-Ctrl-F or |~) for the input file.

13.1.4.5.2 TIBInputDelimitedFile.ReadBlanksAsNull Property

Reads blank spaces in the input file as null characters.

Pascal

```
property ReadBlanksAsNull: Boolean;
```

C++

```
__property Boolean ReadBlanksAsNull;
```

Description

Set ReadBlanksAsNull to true read blank spaces as null characters in the input file.

13.1.4.5.3 TIBInputDelimitedFile.RowDelimiter Property

Sets the row delimiter for the input file.

Pascal

```
property RowDelimiter: string;
```

C++

```
__property UnicodeString RowDelimiter;
```

Description

Use RowDelimiter to set the column delimiter (either Tab-Ctrl-F or |~) for the input file.

13.1.4.5.4 TIBInputDelimitedFile.SkipTitles Property

Skips the first record of a delimited file.

Pascal

```
property SkipTitles: Boolean;
```

C++

```
__property Boolean SkipTitles;
```

Description

Set SkipTitles to true to treat the first record of a delimited file as titles and skip it. Field titles are not useful in batch inputs, and this property allows you to skip them.

13.1.5 TIBInputRawFile Class

TIBInputRawFile inputs data from a raw file.

Class Hierarchy

Pascal

```
TIBInputRawFile = class(TIBBatchInput);
```

C++

```
class TIBInputRawFile : public TIBBatchInput;
```

File

IBSQL

Description

Use a TIBInputRawFile object input data from a raw file. A raw file is the equivalent to InterBase external file output. Raw files are not limited to a straight character format.

13.1.5.1 TIBInputRawFile Members

The following tables list the members exposed by TIBInputRawFile.

Protected Fields

	Name	Description
	FColumns (see page 495)	This is FColumns, a member of class TIBBatch.
	FFile (see page 506)	This is FFile, a member of class TIBInputRawFile.
	FFilename (see page 495)	This is FFilename, a member of class TIBBatch.

	FParams (see page 496)	This is FParams, a member of class TIBBatch.

Public Methods

	Name	Description
	Destroy (see page 506)	Destroys the instance of TIBInputRawFile (see page 505).
	ReadParameters (see page 507)	Reads the XSQLDA input parameters.
	ReadyFile (see page 507)	Prepares the output for the XSQLDA.

Public Properties

	Name	Description
	Columns (see page 496)	Returns the XSQLDA columns.
	Filename (see page 497)	Sets or displays the name of the external file.
	Params (see page 497)	Returns the XSQLDA parameters.

13.1.5.2 TIBInputRawFile.Destroy Destructor

Destroys the instance of TIBInputRawFile (see page 505).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBInputRawFile();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TIBInputRawFile (see page 505) object is not already freed and only then calls Destroy.

13.1.5.3 TIBInputRawFile Fields

The fields of the TIBInputRawFile class are listed here.

Protected Fields

	Name	Description
	FFile (see page 506)	This is FFile, a member of class TIBInputRawFile.

13.1.5.3.1 TIBInputRawFile.FFile Field

Pascal

```
FFile: TFileStream;
```

C++

```
TFileStream FFile;
```

Description

This is FFile, a member of class TIBInputRawFile.

13.1.5.4 TIBInputRawFile Methods

The methods of the TIBInputRawFile class are listed here.

Public Methods

	Name	Description
	ReadParameters (see page 507)	Reads the XSQLDA input parameters.
	ReadyFile (see page 507)	Prepares the output for the XSQLDA.

13.1.5.4.1 TIBInputRawFile.ReadParameters Method

Reads the XSQLDA input parameters.

Pascal

```
function ReadParameters: Boolean; override;
```

C++

```
virtual __fastcall Boolean ReadParameters();
```

Description

Call ReadParameters to read the input parameters of the extended SQL descriptor area (XSQLDA).

13.1.5.4.2 TIBInputRawFile.ReadyFile Method

Prepares the output for the XSQLDA.

Pascal

```
procedure ReadyFile; override;
```

C++

```
virtual __fastcall ReadyFile();
```

Description

Call ReadyFile to prepare the output for the extended SQL descriptor area (XSQLDA).

13.1.6 TIBOutputDelimitedFile Class

TIBOutputDelimitedFile performs batch output of data to delimited files.

Class Hierarchy

Pascal

```
TIBOutputDelimitedFile = class(TIBBatchOutput);
```

C++

```
class TIBOutputDelimitedFile : public TIBBatchOutput;
```

File

IBSQL

Description

Use a TIBOutputDelimitedFile object to perform batch output of data to delimited files.

13.1.6.1 TIBOutputDelimitedFile Members

The following tables list the members exposed by TIBOutputDelimitedFile.

Protected Fields

	Name	Description
	FColDelimiter (see page 509)	This is FColDelimiter, a member of class TIBOutputDelimitedFile.
	FColumns (see page 495)	This is FColumns, a member of class TIBBatch.
	FFile (see page 509)	This is FFile, a member of class TIBOutputDelimitedFile.
	FFilename (see page 495)	This is FFilename, a member of class TIBBatch.
	FOutputTitles (see page 509)	This is FOutputTitles, a member of class TIBOutputDelimitedFile.
	FParams (see page 496)	This is FParams, a member of class TIBBatch.
	FRowDelimiter (see page 509)	This is FRowDelimiter, a member of class TIBOutputDelimitedFile.

Public Methods

	Name	Description
	Destroy (see page 508)	Destroys the instance of TIBOutputDelimitedFile (see page 507).
	ReadyFile (see page 510)	Prepares the output for the XSQLDA.
	WriteColumns (see page 510)	Outputs the data in columns in the XSQLDA.

Public Properties

	Name	Description
	ColDelimiter (see page 510)	Sets the column delimiter for the output file.
	Columns (see page 496)	Returns the XSQLDA columns.
	Filename (see page 497)	Sets or displays the name of the external file.
	OutputTitles (see page 511)	Outputs the titles at the top of the file.
	Params (see page 497)	Returns the XSQLDA parameters.
	RowDelimiter (see page 511)	Sets the row delimiter for the output file.

13.1.6.2 TIBOutputDelimitedFile.Destroy Destructor

Destroys the instance of TIBOutputDelimitedFile (see page 507).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBOutputDelimitedFile();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TIBOutputDelimitedFile (see page 507) object is not already freed and only then calls Destroy.

13.1.6.3 TIBOutputDelimitedFile Fields

The fields of the TIBOutputDelimitedFile class are listed here.

Protected Fields

	Name	Description
	FColDelimiter (see page 509)	This is FColDelimiter, a member of class TIBOutputDelimitedFile.
	FFile (see page 509)	This is FFile, a member of class TIBOutputDelimitedFile.

	FOutputTitles (see page 509)	This is FOutputTitles, a member of class TIBOutputDelimitedFile.
	FRowDelimiter (see page 509)	This is FRowDelimiter, a member of class TIBOutputDelimitedFile.

13.1.6.3.1 TIBOutputDelimitedFile.FColDelimiter Field

Pascal

```
FColDelimiter: string;
```

C++

```
UnicodeString FColDelimiter;
```

Description

This is FColDelimiter, a member of class TIBOutputDelimitedFile.

13.1.6.3.2 TIBOutputDelimitedFile.FFile Field

Pascal

```
FFile: TFileStream;
```

C++

```
TFileStream FFile;
```

Description

This is FFile, a member of class TIBOutputDelimitedFile.

13.1.6.3.3 TIBOutputDelimitedFile.FOutputTitles Field

Pascal

```
FOutputTitles: Boolean;
```

C++

```
Boolean FOutputTitles;
```

Description

This is FOutputTitles, a member of class TIBOutputDelimitedFile.

13.1.6.3.4 TIBOutputDelimitedFile.FRowDelimiter Field

Pascal

```
FRowDelimiter: string;
```

C++

```
UnicodeString FRowDelimiter;
```

Description

This is FRowDelimiter, a member of class TIBOutputDelimitedFile.

13.1.6.4 TIBOutputDelimitedFile Methods

The methods of the TIBOutputDelimitedFile class are listed here.

Public Methods

	Name	Description
	ReadyFile (see page 510)	Prepares the output for the XSQLDA.
	WriteColumns (see page 510)	Outputs the data in columns in the XSQLDA.

13.1.6.4.1 TIBOutputDelimitedFile.ReadyFile Method

Prepares the output for the XSQLDA.

Pascal

```
procedure ReadyFile; override;
```

C++

```
virtual __fastcall ReadyFile();
```

Description

Call Ready file to prepare the output for the extended SQL descriptor area (XSQLDA).

13.1.6.4.2 TIBOutputDelimitedFile.WriteColumns Method

Outputs the data in columns in the XSQLDA.

Pascal

```
function WriteColumns: Boolean; override;
```

C++

```
virtual __fastcall Boolean WriteColumns();
```

Description

Set WriteColumns to true to output data in columns in the extended SQL descriptor area (XSQLDA).

13.1.6.5 TIBOutputDelimitedFile Properties

The properties of the TIBOutputDelimitedFile class are listed here.

Public Properties

	Name	Description
	ColDelimiter (see page 510)	Sets the column delimiter for the output file.
	OutputTitles (see page 511)	Outputs the titles at the top of the file.
	RowDelimiter (see page 511)	Sets the row delimiter for the output file.

13.1.6.5.1 TIBOutputDelimitedFile.ColDelimiter Property

Sets the column delimiter for the output file.

Pascal

```
property ColDelimiter: string;
```

C++

```
__property UnicodeString ColDelimiter;
```

Description

Use ColDelimiter to set the column delimiter (either Tab-Ctrl-F or |~) for the output file.

13.1.6.5.2 TIBOutputDelimitedFile.OutputTitles Property

Outputs the titles at the top of the file.

Pascal

```
property OutputTitles: Boolean;
```

C++

```
__property Boolean OutputTitles;
```

Description

Set OutputTitles to true to output the titles at the top of the file.

13.1.6.5.3 TIBOutputDelimitedFile.RowDelimiter Property

Sets the row delimiter for the output file.

Pascal

```
property RowDelimiter: string;
```

C++

```
__property UnicodeString RowDelimiter;
```

Description

Use RowDelimiter to set the column delimiter (either Tab-Ctrl-F or |~) for the output file.

13.1.7 TIBOutputRawFile Class

TIBOutputRawFile outputs data to a raw file format.

Class Hierarchy

Pascal

```
TIBOutputRawFile = class(TIBBatchOutput);
```

C++

```
class TIBOutputRawFile : public TIBBatchOutput;
```

File

IBSQL

Description

Use a TIBOutputRawFile object to output data from a raw file. A raw file is the equivalent to InterBase external file output. Raw files are not limited to a straight character format.

13.1.7.1 TIBOutputRawFile Members

The following tables list the members exposed by TIBOutputRawFile.

Protected Fields

	Name	Description
	FColumns (see page 495)	This is FColumns, a member of class TIBBatch.
	FFile (see page 512)	This is FFile, a member of class TIBOutputRawFile.
	FFilename (see page 495)	This is FFilename, a member of class TIBBatch.
	FParams (see page 496)	This is FParams, a member of class TIBBatch.

Public Methods

	Name	Description
	Destroy (see page 512)	Destroys the instance of TIBOutputRawFile (see page 511).
	ReadyFile (see page 513)	Prepares the output for the XSQLDA.
	WriteColumns (see page 513)	Outputs the data in columns in the XSQLDA.

Public Properties

	Name	Description
	Columns (see page 496)	Returns the XSQLDA columns.
	Filename (see page 497)	Sets or displays the name of the external file.
	Params (see page 497)	Returns the XSQLDA parameters.

13.1.7.2 TIBOutputRawFile.Destroy Destructor

Destroys the instance of TIBOutputRawFile (see page 511).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBOutputRawFile();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TIBOutputRawFile (see page 511) object is not already freed and only then calls Destroy.

13.1.7.3 TIBOutputRawFile Fields

The fields of the TIBOutputRawFile class are listed here.

Protected Fields

	Name	Description
	FFile (see page 512)	This is FFile, a member of class TIBOutputRawFile.

13.1.7.3.1 TIBOutputRawFile.FFile Field

Pascal

```
FFile: TFileStream;
```

C++

```
TFileStream FFile;
```

Description

This is FFile, a member of class TIBOutputRawFile.

13.1.7.4 TIBOutputRawFile Methods

The methods of the TIBOutputRawFile class are listed here.

Public Methods

	Name	Description
	ReadyFile (see page 513)	Prepares the output for the XSQLDA.
	WriteColumns (see page 513)	Outputs the data in columns in the XSQLDA.

13.1.7.4.1 TIBOutputRawFile.ReadyFile Method

Prepares the output for the XSQLDA.

Pascal

```
procedure ReadyFile; override;
```

C++

```
virtual __fastcall ReadyFile();
```

Description

Call Ready file to prepare the output for the extended SQL descriptor area (XSQLDA).

13.1.7.4.2 TIBOutputRawFile.WriteColumns Method

Outputs the data in columns in the XSQLDA.

Pascal

```
function WriteColumns: Boolean; override;
```

C++

```
virtual __fastcall Boolean WriteColumns();
```

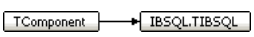
Description

Call WriteColumns to output data in columns in the extended SQL descriptor area (XSQLDA).

13.1.8 TIBSQL Class

TIBSQL provides an object for executing an InterBase SQL (see page 534) statement with minimal overhead.

Class Hierarchy



Pascal

```
TIBSQL = class(TComponent);
```

C++

```
class TIBSQL : public TComponent;
```

File

IBSQL

Description

Use a TIBSQL object to execute an InterBase SQL (see page 534) statement with minimal overhead. TIBSQL has no standard interface to data-aware controls and is unidirectional.

See Also

Getting Started with InterBase Express

13.1.8.1 TIBSQL Members

The following tables list the members exposed by TIBSQL.

Protected Fields

	Name	Description
	FBase (see page 517)	This is FBase, a member of class TIBSQL.
	FBOF (see page 517)	This is FBOF, a member of class TIBSQL.
	FCursor (see page 517)	This is FCursor, a member of class TIBSQL.
	FEOF (see page 517)	This is FEOF, a member of class TIBSQL.
	FGenerateParamNames (see page 518)	This is FGenerateParamNames, a member of class TIBSQL.
	FGoToFirstRecordOnExecute (see page 518)	This is FGoToFirstRecordOnExecute, a member of class TIBSQL.
	FHandle (see page 518)	This is FHandle, a member of class TIBSQL.
	FOnSQLChanging (see page 518)	This is FOnSQLChanging, a member of class TIBSQL.
	FOpen (see page 519)	This is FOpen, a member of class TIBSQL.
	FParamCheck (see page 519)	This is FParamCheck, a member of class TIBSQL.
	FPrepared (see page 519)	This is FPrepared, a member of class TIBSQL.
	FProcessedSQL (see page 519)	This is FProcessedSQL, a member of class TIBSQL.
	FRecordCount (see page 519)	This is FRecordCount, a member of class TIBSQL.
	FSQL (see page 520)	This is FSQL, a member of class TIBSQL.
	FSQLParams (see page 520)	This is FSQLParams, a member of class TIBSQL.
	FSQLRecord (see page 520)	This is FSQLRecord, a member of class TIBSQL.
	FSQLType (see page 520)	This is FSQLType, a member of class TIBSQL.

Protected Methods

	Name	Description
	BeforeTransactionEnd (see page 522)	This is BeforeTransactionEnd, a member of class TIBSQL.
	DoBeforeDatabaseDisconnect (see page 524)	This is DoBeforeDatabaseDisconnect, a member of class TIBSQL.
	GetDatabase (see page 525)	This is GetDatabase, a member of class TIBSQL.
	GetDBHandle (see page 525)	This is GetDBHandle, a member of class TIBSQL.
	GetEOF (see page 525)	This is GetEOF, a member of class TIBSQL.
	GetFieldIndex (see page 525)	This is GetFieldIndex, a member of class TIBSQL.
	GetFields (see page 525)	This is GetFields, a member of class TIBSQL.
	GetPlan (see page 526)	This is GetPlan, a member of class TIBSQL.
	GetRecordCount (see page 526)	This is GetRecordCount, a member of class TIBSQL.
	GetRowsAffected (see page 526)	This is GetRowsAffected, a member of class TIBSQL.
	GetSQLParams (see page 526)	This is GetSQLParams, a member of class TIBSQL.

	GetTransaction (see page 527)	This is GetTransaction, a member of class TIBSQL.
	GetTRHandle (see page 527)	This is GetTRHandle, a member of class TIBSQL.
	PreprocessSQL (see page 528)	This is PreprocessSQL, a member of class TIBSQL.
	SetDatabase (see page 528)	This is SetDatabase, a member of class TIBSQL.
	SetSQL (see page 528)	This is SetSQL, a member of class TIBSQL.
	SetTransaction (see page 529)	This is SetTransaction, a member of class TIBSQL.
	SQLChanging (see page 529)	This is SQLChanging, a member of class TIBSQL.

Public Methods

	Name	Description
	BatchInput (see page 521)	Executes the parameterized query in SQL (see page 534) for input in the referenced input object.
	BatchOutput (see page 522)	Outputs the selected query in SQL (see page 534) to the referenced OutputObject.
	Call (see page 522)	Returns an error message based on the error code.
	CheckClosed (see page 522)	raises an exception if the query is not closed.
	CheckOpen (see page 523)	raises an exception if the query is not open.
	CheckValidStatement (see page 523)	raises an exception if the query does not have a valid statement.
	Close (see page 523)	Closes the query.
V	Create (see page 516)	Creates an instance of a TIBSQL (see page 513) component.
	Current (see page 523)	Returns an extended SQL (see page 534) descriptor for the current record.
V	Destroy (see page 516)	Frees all resources associated with this instance.
	ExecQuery (see page 524)	Executes an SQL (see page 534) query.
	FieldByName (see page 524)	Returns the XSQLVAR fields by name.
	FreeHandle (see page 524)	Frees InterBase resources associated with the query.
	GetUniqueRelationName (see page 527)	Gets the unique relation name.
	Next (see page 527)	Returns an extended SQL (see page 534) descriptor for the next record.
	ParamByName (see page 527)	Accesses parameter information based on a specified parameter name.
	Prepare (see page 528)	Prepares a query for execution.

Public Properties

	Name	Description
R	Bof (see page 530)	Indicates whether or not a query is at the beginning of the dataset.
R	DBHandle (see page 530)	Specifies the database handle for the query.
R	Eof (see page 530)	Indicates whether or not a query is at the end of the dataset.
R	FieldIndex (see page 531)	Returns the index of the named field.
R	Fields (see page 531)	Returns the XSQLVAR fields.
	GenerateParamNames (see page 531)	Generates a list of parameter names for the query.
R	Handle (see page 532)	Specifies the handle for the query.
R	Open (see page 532)	Determines if the dataset is open.
R	Params (see page 533)	Returns the XSQLDA parameters.
R	Plan (see page 533)	Returns the plan for the query.
R	Prepared (see page 533)	Indicates whether or not the query has been prepared.
R	RecordCount (see page 533)	Returns the current count of records from the query.
R	RowsAffected (see page 534)	Returns the number of rows affected.
R	SQLType (see page 534)	Returns the type of query to be executed.
R	TRHandle (see page 535)	Specifies the transaction handle for the query.
R	UniqueRelationName (see page 535)	Indicates the unique relation name.

Published Events

	Name	Description
	OnSQLChanging (see page 535)	Occurs when the SQL (see page 534) query is being modified.

Published Properties

	Name	Description
	Database (see page 530)	Sets or returns the database associated with the query.
	GoToFirstRecordOnExecute (see page 532)	Goes to the first record in the result set upon opening it.
	ParamCheck (see page 532)	Specifies whether the parameter list for an SQL (see page 534) query is regenerated if the SQL (see page 534) property changes at runtime.
	SQL (see page 534)	Sets the SQL query to be executed.
	Transaction (see page 535)	Sets or returns the transaction to be used by the query.

13.1.8.2 TIBSQL.Create Constructor

Creates an instance of a TIBSQL (see page 513) component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBSQL(TComponent * AOwner);
```

Description

Call (see page 522) Create to create an instance of a TIBSQL (see page 513) component.

13.1.8.3 TIBSQL.Destroy Destructor

Frees all resources associated with this instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBSQL();
```

Description

Do not call Destroy directly in an application. Usually destruction of objects is handled automatically by Delphi. If an application creates its own instance of an update object, however, the application should call Free, which verifies that the update object is not already freed before calling Destroy.

13.1.8.4 TIBSQL Fields

The fields of the TIBSQL class are listed here.

Protected Fields

	Name	Description
	FBase (see page 517)	This is FBase, a member of class TIBSQL.
	FBOF (see page 517)	This is FBOF, a member of class TIBSQL.
	FCursor (see page 517)	This is FCursor, a member of class TIBSQL.
	FEOF (see page 517)	This is FEOF, a member of class TIBSQL.
	FGenerateParamNames (see page 518)	This is FGenerateParamNames, a member of class TIBSQL.
	FGoToFirstRecordOnExecute (see page 518)	This is FGoToFirstRecordOnExecute, a member of class TIBSQL.
	FHandle (see page 518)	This is FHandle, a member of class TIBSQL.
	FOnSQLChanging (see page 518)	This is FOnSQLChanging, a member of class TIBSQL.

	FOpen (see page 519)	This is FOpen, a member of class TIBSQL.
	FParamCheck (see page 519)	This is FParamCheck, a member of class TIBSQL.
	FPrepared (see page 519)	This is FPrepared, a member of class TIBSQL.
	FProcessedSQL (see page 519)	This is FProcessedSQL, a member of class TIBSQL.
	FRecordCount (see page 519)	This is FRecordCount, a member of class TIBSQL.
	FSQL (see page 520)	This is FSQL, a member of class TIBSQL.
	FSQLParams (see page 520)	This is FSQLParams, a member of class TIBSQL.
	FSQLRecord (see page 520)	This is FSQLRecord, a member of class TIBSQL.
	FSQLType (see page 520)	This is FSQLType, a member of class TIBSQL.

13.1.8.4.1 TIBSQL.FBase Field

Pascal

```
FBase: TIBBase;
```

C++

```
TIBBase FBase;
```

Description

This is FBase, a member of class TIBSQL.

13.1.8.4.2 TIBSQL.FBOF Field

Pascal

```
FBOF: Boolean;
```

C++

```
Boolean FBOF;
```

Description

This is FBOF, a member of class TIBSQL.

13.1.8.4.3 TIBSQL.FCursor Field

Pascal

```
FCursor: String;
```

C++

```
UnicodeString FCursor;
```

Description

This is FCursor, a member of class TIBSQL.

13.1.8.4.4 TIBSQL.FEOF Field

Pascal

```
FEOF: Boolean;
```

C++

```
Boolean FEOF;
```

Description

This is FEOF, a member of class TIBSQL.

13.1.8.4.5 TIBSQL.FGenerateParamNames Field

Pascal

```
FGenerateParamNames: Boolean;
```

C++

```
Boolean FGenerateParamNames;
```

Description

This is FGenerateParamNames, a member of class TIBSQL.

13.1.8.4.6 TIBSQL.FGoToFirstRecordOnExecute Field

Pascal

```
FGoToFirstRecordOnExecute: Boolean;
```

C++

```
Boolean FGoToFirstRecordOnExecute;
```

Description

This is FGoToFirstRecordOnExecute, a member of class TIBSQL.

13.1.8.4.7 TIBSQL.FHandle Field

Pascal

```
FHandle: TISC_STMT_HANDLE;
```

C++

```
TISC_STMT_HANDLE FHandle;
```

Description

This is FHandle, a member of class TIBSQL.

13.1.8.4.8 TIBSQL.FOnSQLChanging Field

Pascal

```
FOnSQLChanging: TNotifyEvent;
```

C++

```
TNotifyEvent FOnSQLChanging;
```

Description

This is FOnSQLChanging, a member of class TIBSQL.

13.1.8.4.9 TIBSQL.FOpen Field

Pascal

```
FOpen: Boolean;
```

C++

```
Boolean FOpen;
```

Description

This is FOpen, a member of class TIBSQL.

13.1.8.4.10 TIBSQL.FParamCheck Field

Pascal

```
FParamCheck: Boolean;
```

C++

```
Boolean FParamCheck;
```

Description

This is FParamCheck, a member of class TIBSQL.

13.1.8.4.11 TIBSQL.FPrepared Field

Pascal

```
FPrepared: Boolean;
```

C++

```
Boolean FPrepared;
```

Description

This is FPrepared, a member of class TIBSQL.

13.1.8.4.12 TIBSQL.FProcessedSQL Field

Pascal

```
FProcessedSQL: TStrings;
```

C++

```
TStrings FProcessedSQL;
```

Description

This is FProcessedSQL, a member of class TIBSQL.

13.1.8.4.13 TIBSQL.FRecordCount Field

Pascal

```
FRecordCount: Integer;
```

C++

```
int FRecordCount;
```

Description

This is FRecordCount, a member of class TIBSQL.

13.1.8.4.14 TIBSQL.FSQL Field

Pascal

```
FSQL: TStrings;
```

C++

```
TStrings FSQL;
```

Description

This is FSQL, a member of class TIBSQL.

13.1.8.4.15 TIBSQL.FSQLParams Field

Pascal

```
FSQLParams: TIBXSQLDA;
```

C++

```
TIBXSQLDA FSQLParams;
```

Description

This is FSQLParams, a member of class TIBSQL.

13.1.8.4.16 TIBSQL.FSQLRecord Field

Pascal

```
FSQLRecord: TIBXSQLDA;
```

C++

```
TIBXSQLDA FSQLRecord;
```

Description

This is FSQLRecord, a member of class TIBSQL.

13.1.8.4.17 TIBSQL.FSQLType Field

Pascal

```
FSQLType: TIBSQLTypes;
```

C++

```
TIBSQLTypes FSQLType;
```

Description

This is FSQLType, a member of class TIBSQL.

13.1.8.5 TIBSQL Methods

The methods of the TIBSQL class are listed here.

Protected Methods

	Name	Description
	BeforeTransactionEnd (see page 522)	This is BeforeTransactionEnd, a member of class TIBSQL.
	DoBeforeDatabaseDisconnect (see page 524)	This is DoBeforeDatabaseDisconnect, a member of class TIBSQL.
	GetDatabase (see page 525)	This is GetDatabase, a member of class TIBSQL.
	GetDBHandle (see page 525)	This is GetDBHandle, a member of class TIBSQL.
	GetEOF (see page 525)	This is GetEOF, a member of class TIBSQL.
	GetFieldIndex (see page 525)	This is GetFieldIndex, a member of class TIBSQL.
	GetFields (see page 525)	This is GetFields, a member of class TIBSQL.
	GetPlan (see page 526)	This is GetPlan, a member of class TIBSQL.
	GetRecordCount (see page 526)	This is GetRecordCount, a member of class TIBSQL.
	GetRowsAffected (see page 526)	This is GetRowsAffected, a member of class TIBSQL.
	GetSQLParams (see page 526)	This is GetSQLParams, a member of class TIBSQL.
	GetTransaction (see page 527)	This is GetTransaction, a member of class TIBSQL.
	GetTRHandle (see page 527)	This is GetTRHandle, a member of class TIBSQL.
	PreprocessSQL (see page 528)	This is PreprocessSQL, a member of class TIBSQL.
	SetDatabase (see page 528)	This is SetDatabase, a member of class TIBSQL.
	SetSQL (see page 528)	This is SetSQL, a member of class TIBSQL.
	SetTransaction (see page 529)	This is SetTransaction, a member of class TIBSQL.
	SQLChanging (see page 529)	This is SQLChanging, a member of class TIBSQL.

Public Methods

	Name	Description
	BatchInput (see page 521)	Executes the parameterized query in SQL (see page 534) for input in the referenced input object.
	BatchOutput (see page 522)	Outputs the selected query in SQL (see page 534) to the referenced OutputObject.
	Call (see page 522)	Returns an error message based on the error code.
	CheckClosed (see page 522)	raises an exception if the query is not closed.
	CheckOpen (see page 523)	raises an exception if the query is not open.
	CheckValidStatement (see page 523)	raises an exception if the query does not have a valid statement.
	Close (see page 523)	Closes the query.
	Current (see page 523)	Returns an extended SQL (see page 534) descriptor for the current record.
	ExecQuery (see page 524)	Executes an SQL (see page 534) query.
	FieldByName (see page 524)	Returns the XSQLVAR fields by name.
	FreeHandle (see page 524)	Frees InterBase resources associated with the query.
	GetUniqueRelationName (see page 527)	Gets the unique relation name.
	Next (see page 527)	Returns an extended SQL (see page 534) descriptor for the next record.
	ParamByName (see page 527)	Accesses parameter information based on a specified parameter name.
	Prepare (see page 528)	Prepares a query for execution.

13.1.8.5.1 TIBSQL.BatchInput Method

Executes the parameterized query in SQL (see page 534) for input in the referenced input object.

Pascal

```
procedure BatchInput(InputObject: TIBBatchInput);
```

C++

```
__fastcall BatchInput(TIBBatchInput InputObject);
```

Description

Call (see page 522) BatchInput to execute the parameterized query in SQL (see page 534) for input in the referenced input object.

13.1.8.5.2 TIBSQL.BatchOutput Method

Outputs the selected query in SQL (see page 534) to the referenced OutputObject.

Pascal

```
procedure BatchOutput(OutputObject: TIBBatchOutput);
```

C++

```
__fastcall BatchOutput(TIBBatchOutput OutputObject);
```

Description

Call (see page 522) BatchOutput to output the selected query in SQL (see page 534) to the referenced OutputObject.

13.1.8.5.3 TIBSQL.BeforeTransactionEnd Method

Pascal

```
procedure BeforeTransactionEnd(Sender: TObject);
```

C++

```
__fastcall BeforeTransactionEnd(TObject * Sender);
```

Description

This is BeforeTransactionEnd, a member of class TIBSQL.

13.1.8.5.4 TIBSQL.Call Method

Returns an error message based on the error code.

Pascal

```
function Call(ErrCode: ISC_STATUS; RaiseError: Boolean): ISC_STATUS;
```

C++

```
__fastcall ISC_STATUS Call(ISC_STATUS ErrCode, Boolean RaiseError);
```

Description

Call is an internal method used to make calls to the InterBase API, and gives you the option of raising an exception or returning an error based on the value of RaiseError.

13.1.8.5.5 TIBSQL.CheckClosed Method

raises an exception if the query is not closed.

Pascal

```
procedure CheckClosed;
```

C++

```
__fastcall CheckClosed();
```

Description

Call (see page 522) CheckClosed to raise an exception if the query is not closed.

13.1.8.5.6 TIBSQL.CheckOpen Method

raises an exception if the query is not open.

Pascal

```
procedure CheckOpen;
```

C++

```
__fastcall CheckOpen();
```

Description

Call (see page 522) CheckOpen to raise an exception if the query is closed.

13.1.8.5.7 TIBSQL.CheckValidStatement Method

raises an exception if the query does not have a valid statement.

Pascal

```
procedure CheckValidStatement;
```

C++

```
__fastcall CheckValidStatement();
```

Description

Call (see page 522) CheckValidStatement to raise an exception if the query does not have a valid statement.

13.1.8.5.8 TIBSQL.Close Method

Closes the query.

Pascal

```
procedure Close;
```

C++

```
__fastcall Close();
```

Description

Call (see page 522) Close to close the query.

13.1.8.5.9 TIBSQL.Current Method

Returns an extended SQL (see page 534) descriptor for the current record.

Pascal

```
function Current: TIBXSQLDA;
```

C++

```
__fastcall TIBXSQLDA Current();
```

Description

Call (see page 522) Current to get an extended SQL (see page 534) descriptor for the current record.

13.1.8.5.10 TIBSQL.DoBeforeDatabaseDisconnect Method

Pascal

```
procedure DoBeforeDatabaseDisconnect(Sender: TObject);
```

C++

```
__fastcall DoBeforeDatabaseDisconnect(TObject * Sender);
```

Description

This is DoBeforeDatabaseDisconnect, a member of class TIBSQL.

13.1.8.5.11 TIBSQL.ExecQuery Method

Executes an SQL (see page 534) query.

Pascal

```
procedure ExecQuery;
```

C++

```
__fastcall ExecQuery();
```

Description

Note: Call (see page 522) ExecQuery to execute the SQL (see page 534) query.

13.1.8.5.12 TIBSQL.FieldByName Method

Returns the XSQLVAR fields by name.

Pascal

```
function FieldByName(FieldName: String): TIBXSQLVAR;
```

C++

```
__fastcall TIBXSQLVAR FieldByName(UnicodeString FieldName);
```

Description

Use the FieldByName method to return the XSQLVAR fields by name.

13.1.8.5.13 TIBSQL.FreeHandle Method

Frees InterBase resources associated with the query.

Pascal

```
procedure FreeHandle;
```

C++

```
__fastcall FreeHandle();
```

Description

Call (see page 522) FreeHandle to free the InterBase resources associated with the query.

13.1.8.5.14 TIBSQL.GetDatabase Method

Pascal

```
function GetDatabase: TIBDatabase;
```

C++

```
__fastcall TIBDatabase GetDatabase();
```

Description

This is GetDatabase, a member of class TIBSQL.

13.1.8.5.15 TIBSQL.GetDBHandle Method

Pascal

```
function GetDBHandle: PISC_DB_HANDLE;
```

C++

```
__fastcall PISC_DB_HANDLE GetDBHandle();
```

Description

This is GetDBHandle, a member of class TIBSQL.

13.1.8.5.16 TIBSQL.GetEOF Method

Pascal

```
function GetEOF: Boolean;
```

C++

```
__fastcall Boolean GetEOF();
```

Description

This is GetEOF, a member of class TIBSQL.

13.1.8.5.17 TIBSQL.GetFieldIndex Method

Pascal

```
function GetFieldIndex(FieldName: String): Integer;
```

C++

```
__fastcall int GetFieldIndex(UnicodeString FieldName);
```

Description

This is GetFieldIndex, a member of class TIBSQL.

13.1.8.5.18 TIBSQL.GetFields Method

Pascal

```
function GetFields(const Idx: Integer): TIBXSQLVAR;
```

C++

```
__fastcall TIBXSQLVAR GetFields(const int Idx);
```

Description

This is GetFields, a member of class TIBSQL.

13.1.8.5.19 TIBSQL.GetPlan Method

Pascal

```
function GetPlan: String;
```

C++

```
__fastcall UnicodeString GetPlan();
```

Description

This is GetPlan, a member of class TIBSQL.

13.1.8.5.20 TIBSQL.GetRecordCount Method

Pascal

```
function GetRecordCount: Integer;
```

C++

```
__fastcall int GetRecordCount();
```

Description

This is GetRecordCount, a member of class TIBSQL.

13.1.8.5.21 TIBSQL.GetRowsAffected Method

Pascal

```
function GetRowsAffected: Integer;
```

C++

```
__fastcall int GetRowsAffected();
```

Description

This is GetRowsAffected, a member of class TIBSQL.

13.1.8.5.22 TIBSQL.GetSQLParams Method

Pascal

```
function GetSQLParams: TIBXSQLDA;
```

C++

```
__fastcall TIBXSQLDA GetSQLParams();
```

Description

This is GetSQLParams, a member of class TIBSQL.

13.1.8.5.23 TIBSQL.GetTransaction Method

Pascal

```
function GetTransaction: TIBTransaction;
```

C++

```
__fastcall TIBTransaction GetTransaction();
```

Description

This is GetTransaction, a member of class TIBSQL.

13.1.8.5.24 TIBSQL.GetTRHandle Method

Pascal

```
function GetTRHandle: PISC_TR_HANDLE;
```

C++

```
__fastcall PISC_TR_HANDLE GetTRHandle();
```

Description

This is GetTRHandle, a member of class TIBSQL.

13.1.8.5.25 TIBSQL.GetUniqueRelationName Method

Gets the unique relation name.

Pascal

```
function GetUniqueRelationName: String;
```

C++

```
__fastcall UnicodeString GetUniqueRelationName();
```

Description

Note: Call (see page 522) GetUniqueRelationName to get the unique relation name of the query.

13.1.8.5.26 TIBSQL.Next Method

Returns an extended SQL (see page 534) descriptor for the next record.

Pascal

```
function Next: TIBXSQLDA;
```

C++

```
__fastcall TIBXSQLDA Next();
```

Description

Call (see page 522) Next to get an extended SQL (see page 534) descriptor for the next record.

13.1.8.5.27 TIBSQL.ParamByName Method

Accesses parameter information based on a specified parameter name.

Pascal

```
function ParamByName(Idx: String): TIBXSQLVAR;
```

C++

```
__fastcall TIBXSQLVAR ParamByName(UnicodeString Idx);
```

Description

Call (see page 522) ParamByName to set or use parameter information for a specific parameter based on its name.

Idx is the name of the parameter for which to retrieve information.

13.1.8.5.28 TIBSQL.Prepare Method

Prepares a query for execution.

Pascal

```
procedure Prepare;
```

C++

```
__fastcall Prepare();
```

Description

Call (see page 522) Prepare to prepare a query for execution.

13.1.8.5.29 TIBSQL.PreprocessSQL Method

Pascal

```
procedure PreprocessSQL;
```

C++

```
__fastcall PreprocessSQL();
```

Description

This is PreprocessSQL, a member of class TIBSQL.

13.1.8.5.30 TIBSQL.SetDatabase Method

Pascal

```
procedure SetDatabase(Value: TIBDatabase);
```

C++

```
__fastcall SetDatabase(TIBDatabase Value);
```

Description

This is SetDatabase, a member of class TIBSQL.

13.1.8.5.31 TIBSQL.SetSQL Method

Pascal

```
procedure SetSQL(Value: TStrings);
```

C++

```
__fastcall SetSQL(TStrings Value);
```

Description

This is SetSQL, a member of class TIBSQL.

13.1.8.5.32 TIBSQL.SetTransaction Method

Pascal

```
procedure SetTransaction(Value: TIBTransaction);
```

C++

```
__fastcall SetTransaction(TIBTransaction Value);
```

Description

This is SetTransaction, a member of class TIBSQL.

13.1.8.5.33 TIBSQL.SQLChanging Method

Pascal

```
procedure SQLChanging(Sender: TObject);
```

C++

```
__fastcall SQLChanging(TObject * Sender);
```

Description

This is SQLChanging, a member of class TIBSQL.

13.1.8.6 TIBSQL Properties

The properties of the TIBSQL class are listed here.

Public Properties

	Name	Description
R	Bof (see page 530)	Indicates whether or not a query is at the beginning of the dataset.
R	DBHandle (see page 530)	Specifies the database handle for the query.
R	Eof (see page 530)	Indicates whether or not a query is at the end of the dataset.
R	FieldIndex (see page 531)	Returns the index of the named field.
R	Fields (see page 531)	Returns the XSQLVAR fields.
	GenerateParamNames (see page 531)	Generates a list of parameter names for the query.
R	Handle (see page 532)	Specifies the handle for the query.
R	Open (see page 532)	Determines if the dataset is open.
R	Params (see page 533)	Returns the XSQLDA parameters.
R	Plan (see page 533)	Returns the plan for the query.
R	Prepared (see page 533)	Indicates whether or not the query has been prepared.
R	RecordCount (see page 533)	Returns the current count of records from the query.
R	RowsAffected (see page 534)	Returns the number of rows affected.
R	SQLType (see page 534)	Returns the type of query to be executed.
R	TRHandle (see page 535)	Specifies the transaction handle for the query.
R	UniqueRelationName (see page 535)	Indicates the unique relation name.

Published Properties

	Name	Description
	Database (see page 530)	Sets or returns the database associated with the query.
	GoToFirstRecordOnExecute (see page 532)	Goes to the first record in the result set upon opening it.
	ParamCheck (see page 532)	Specifies whether the parameter list for an SQL (see page 534) query is regenerated if the SQL (see page 534) property changes at runtime.
	SQL (see page 534)	Sets the SQL query to be executed.
	Transaction (see page 535)	Sets or returns the transaction to be used by the query.

13.1.8.6.1 TIBSQL.Bof Property

Indicates whether or not a query is at the beginning of the dataset.

Pascal

```
property Bof: Boolean;
```

C++

```
__property Boolean Bof;
```

Description

Use the BOF property to determine whether or not a query is at the beginning of the dataset.

13.1.8.6.2 TIBSQL.Database Property

Sets or returns the database associated with the query.

Pascal

```
property Database: TIBDatabase;
```

C++

```
__property TIBDatabase Database;
```

Description

Use the Database property to set or return the database associated with the query.

13.1.8.6.3 TIBSQL.DBHandle Property

Specifies the database handle for the query.

Pascal

```
property DBHandle: PISC_DB_HANDLE;
```

C++

```
__property PISC_DB_HANDLE DBHandle;
```

Description

Use the DBHandle property to return the database handle for the query.

13.1.8.6.4 TIBSQL.Eof Property

Indicates whether or not a query is at the end of the dataset.

Pascal

```
property Eof: Boolean;
```

C++

```
__property Boolean Eof;
```

Description

Use the EOF property to determine whether or not a query is at the end of the dataset and whether or not the query returned any result rows.

In addition, if this property is true immediately after the query is opened, then it means that the dataset is empty.

13.1.8.6.5 TIBSQL.FieldIndex Property

Returns the index of the named field.

Pascal

```
property FieldIndex [FieldName: String]: Integer;
```

C++

```
__property int FieldIndex[UnicodeString FieldName];
```

Description

Use the FieldIndex property to return the index of the name field.

13.1.8.6.6 TIBSQL.Fields Property

Returns the XSQLVAR fields.

Pascal

```
property Fields [const Idx: Integer]: TIBXSQLVAR;
```

C++

```
__property TIBXSQLVAR Fields[int const Idx];
```

Description

Use the Fields property to return the XSQLVAR fields.

13.1.8.6.7 TIBSQL.GenerateParamNames Property

Generates a list of parameter names for the query.

Pascal

```
property GenerateParamNames: Boolean;
```

C++

```
__property Boolean GenerateParamNames;
```

Description

Set GenerateParamNames to true to have the query generate a list of parameter names.

13.1.8.6.8 TIBSQL.GoToFirstRecordOnExecute Property

Goes to the first record in the result set upon opening it.

Pascal

```
property GoToFirstRecordOnExecute: Boolean;
```

C++

```
__property Boolean GoToFirstRecordOnExecute;
```

Description

Use the GoToFirstRecordOnExecute property to go to the first record in a result set upon opening it. By default, this property is set to true. GoToFirstRecordOnExecute exists primarily for use in TIBDataSet, which sets this value to false for its internal TIBSQLs.

13.1.8.6.9 TIBSQL.Handle Property

Specifies the handle for the query.

Pascal

```
property Handle: TISC_STMT_HANDLE;
```

C++

```
__property TISC_STMT_HANDLE Handle;
```

Description

Use the Handle property to get the query handle.

13.1.8.6.10 TIBSQL.Open Property

Determines if the dataset is open.

Pascal

```
property Open: Boolean;
```

C++

```
__property Boolean Open;
```

Description

Use the Open property to determine if the dataset is open.

13.1.8.6.11 TIBSQL.ParamCheck Property

Specifies whether the parameter list for an SQL (see page 534) query is regenerated if the SQL (see page 534) property changes at runtime.

Pascal

```
property ParamCheck: Boolean;
```

C++

```
__property Boolean ParamCheck;
```

Description

This property is useful for data definition language (DDL) statements that contain parameters as part of the DDL statement and that are not parameters for the TIBSQL (see page 513) query. For example, the DDL statement to create a stored procedure may contain parameter statements that are part of the stored procedure. Set ParamCheck to false to prevent these parameters from being mistaken for parameters of the TIBSQL (see page 513) query executing the DDL statement.

An application that does not use parameterized queries may choose to set ParamCheck to false, but otherwise ParamCheck should be true.

13.1.8.6.12 TIBSQL.Params Property

Returns the XSQLDA parameters.

Pascal

```
property Params: TIBXSQLDA;
```

C++

```
__property TIBXSQLDA Params;
```

Description

Use the Params property to return the XSQLDA parameters.

13.1.8.6.13 TIBSQL.Plan Property

Returns the plan for the query.

Pascal

```
property Plan: String;
```

C++

```
__property UnicodeString Plan;
```

Description

Use the Plan property to view the query plan once the query has been prepared.

13.1.8.6.14 TIBSQL.Prepared Property

Indicates whether or not the query has been prepared.

Pascal

```
property Prepared: Boolean;
```

C++

```
__property Boolean Prepared;
```

Description

Use the Prepared property to determine whether or not a query has yet been prepared.

13.1.8.6.15 TIBSQL.RecordCount Property

Returns the current count of records from the query.

Pascal

```
property RecordCount: Integer;
```

C++

```
__property int RecordCount;
```

Description

Use the RecordCount property to see how many records are returned by a query. If the result set is to return 100 rows, RecordCount will only be 100 after all the records have been visited. That is, after looking at the first record, RecordCount is 1, and so forth.

13.1.8.6.16 TIBSQL.RowsAffected Property

Returns the number of rows affected.

Pascal

```
property RowsAffected: Integer;
```

C++

```
__property int RowsAffected;
```

Description

Use the RowsAffected property to return the number of rows affected by the query. This property is useful for INSERT, DELETE, and UPDATE statements.

13.1.8.6.17 TIBSQL.SQL Property

Sets the SQL query to be executed.

Pascal

```
property SQL: TStrings;
```

C++

```
__property TStrings SQL;
```

Description

Use the SQL property to write or view the SQL query to be executed.

13.1.8.6.18 TIBSQL.SQLType Property

Returns the type of query to be executed.

Pascal

```
property SQLType: TIBSQLTypes;
```

C++

```
__property TIBSQLTypes SQLType;
```

Description

Use the SQLType to determine the type of query to be executed.

13.1.8.6.19 TIBSQL.Transaction Property

Sets or returns the transaction to be used by the query.

Pascal

```
property Transaction: TIBTransaction;
```

C++

```
__property TIBTransaction Transaction;
```

Description

Use the Transaction property to set or return the transaction to be used by the query.

13.1.8.6.20 TIBSQL.TRHandle Property

Specifies the transaction handle for the query.

Pascal

```
property TRHandle: PISC_TR_HANDLE;
```

C++

```
__property PISC_TR_HANDLE TRHandle;
```

Description

Use the TRHandle property to return the transaction handle for the query.

13.1.8.6.21 TIBSQL.UniqueRelationName Property

Indicates the unique relation name.

Pascal

```
property UniqueRelationName: String;
```

C++

```
__property UnicodeString UniqueRelationName;
```

Description

Use the UniqueRelationName property to indicate the unique relation name for a query that involves only one base table.

13.1.8.7 TIBSQL Events

The events of the TIBSQL class are listed here.

Published Events

	Name	Description
	OnSQLChanging (see page 535)	Occurs when the SQL (see page 534) query is being modified.

13.1.8.7.1 TIBSQL.OnSQLChanging Event

Occurs when the SQL (see page 534) query is being modified.

Pascal

```
property OnSQLChanging: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnSQLChanging;
```

Description

Write an OnSQLChanging event handler to take specific actions when a query is being modified. If an exception is raised in this event, the query is not changed.

13.1.9 TIBXSQLDA Class

TIBXSQLDA provides properties and methods for use with the IBSQL (see page 493) component.

Class Hierarchy

Pascal

```
TIBXSQLDA = class(TObject);
```

C++

```
class TIBXSQLDA : public TObject;
```

File

IBSQL

Description

Use a TIBXSQLDA object to provide properties and methods for use with the IBSQL (see page 493) component.

All Dynamic SQL (DSQL) applications must declare one or more extended descriptor areas (XSQLDAs). The XSQLDA is a host language data structure that DSQL uses to transport data to or from a database when processing an SQL statement string. There are two types of XSQLDAs: input descriptors and output descriptors. Both input and output descriptors are implemented using the XSQLDA structure.

13.1.9.1 TIBXSQLDA Members

The following tables list the members exposed by TIBXSQLDA.

Protected Fields

	Name	Description
	FCount (see page 538)	This is FCount, a member of class TIBXSQLDA.
	FNames (see page 538)	This is FNames, a member of class TIBXSQLDA.
	FSize (see page 538)	This is FSize, a member of class TIBXSQLDA.
	FSQL (see page 539)	This is FSQL, a member of class TIBXSQLDA.
	FUniqueRelationName (see page 539)	This is FUniqueRelationName, a member of class TIBXSQLDA.
	FXSQLDA (see page 539)	This is FXSQLDA, a member of class TIBXSQLDA.
	FXSQLVARs (see page 539)	This is FXSQLVARs, a member of class TIBXSQLDA.

Protected Methods

	Name	Description
	GetModified (see page 540)	This is GetModified, a member of class TIBXSQLDA.
	GetNames (see page 541)	This is GetNames, a member of class TIBXSQLDA.
	GetRecordSize (see page 541)	This is GetRecordSize, a member of class TIBXSQLDA.
	GetXSQLDA (see page 541)	This is GetXSQLDA, a member of class TIBXSQLDA.
	GetXSQLVAR (see page 541)	This is GetXSQLVAR, a member of class TIBXSQLDA.
	GetXSQLVARByName (see page 541)	This is GetXSQLVARByName, a member of class TIBXSQLDA.
	Initialize (see page 542)	This is Initialize, a member of class TIBXSQLDA.
	SetCount (see page 542)	This is SetCount, a member of class TIBXSQLDA.

Public Methods

	Name	Description
	AddName (see page 540)	Adds a name to the XSQLDA structure.
	ByName (see page 540)	Returns the XSQLVAR fields by name.
	Create (see page 537)	Creates an instance of an XSQLDA structure.
V	Destroy (see page 537)	Destroys the XSQLDA structure.

Public Properties

	Name	Description
R	AsXSQLDA (see page 542)	Represents the XSQLDA field's value as an XSQLDA value.
	Count (see page 543)	Returns the number of XSQLDA fields.
R	Modified (see page 543)	Indicates whether a field has been modified.
R	Names (see page 543)	Returns the XSQLDA field names.
R	RecordSize (see page 543)	Returns the size of the XSQLDA record.
R	UniqueRelationName (see page 544)	Returns the name of the unique relation.
R	Vars (see page 544)	Returns the XSQLVAR defined for the XSQLDA parameter.

13.1.9.2 TIBXSQLDA.Create Constructor

Creates an instance of an XSQLDA structure.

Pascal

```
constructor Create(Query: TIBSQL);
```

C++

```
__fastcall TIBXSQLDA(TIBSQL Query);
```

Description

Call Create to create an instance of an XSQLDA structure. Create queries the DynamicSQL component for the structure members.

13.1.9.3 TIBXSQLDA.Destroy Destructor

Destroys the XSQLDA structure.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBXSQLDA();
```

Description

Do not call Destroy directly. Call Free instead. Free checks to ensure that the object instance is not nil before calling Destroy.

13.1.9.4 TIBXSQLDA Fields

The fields of the TIBXSQLDA class are listed here.

Protected Fields

	Name	Description
	FCount (see page 538)	This is FCount, a member of class TIBXSQLDA.
	FNames (see page 538)	This is FNames, a member of class TIBXSQLDA.
	FSize (see page 538)	This is FSize, a member of class TIBXSQLDA.
	FSQL (see page 539)	This is FSQL, a member of class TIBXSQLDA.
	FUniqueRelationName (see page 539)	This is FUniqueRelationName, a member of class TIBXSQLDA.
	FXSQLDA (see page 539)	This is FXSQLDA, a member of class TIBXSQLDA.
	FXSQLVARs (see page 539)	This is FXSQLVARs, a member of class TIBXSQLDA.

13.1.9.4.1 TIBXSQLDA.FCount Field

Pascal

```
FCount: Integer;
```

C++

```
int FCount;
```

Description

This is FCount, a member of class TIBXSQLDA.

13.1.9.4.2 TIBXSQLDA.FNames Field

Pascal

```
FNames: TStrings;
```

C++

```
TStrings FNames;
```

Description

This is FNames, a member of class TIBXSQLDA.

13.1.9.4.3 TIBXSQLDA.FSize Field

Pascal

```
FSize: Integer;
```

C++

```
int FSize;
```

Description

This is FSize, a member of class TIBXSQLDA.

13.1.9.4.4 TIBXSQLDA.FSQL Field

Pascal

```
FSQL: TIBSQL;
```

C++

```
TIBSQL FSQL;
```

Description

This is FSQL, a member of class TIBXSQLDA.

13.1.9.4.5 TIBXSQLDA.FUniqueRelationName Field

Pascal

```
FUniqueRelationName: String;
```

C++

```
UnicodeString FUniqueRelationName;
```

Description

This is FUniqueRelationName, a member of class TIBXSQLDA.

13.1.9.4.6 TIBXSQLDA.FXSQLDA Field

Pascal

```
FXSQLDA: PXSQLDA;
```

C++

```
PXSQLDA FXSQLDA;
```

Description

This is FXSQLDA, a member of class TIBXSQLDA.

13.1.9.4.7 TIBXSQLDA.FXSQLVARs Field

Pascal

```
FXSQLVARs: TIBXSQLVARArray;
```

C++

```
TIBXSQLVARArray FXSQLVARs;
```

Description

This is FXSQLVARs, a member of class TIBXSQLDA.

13.1.9.5 TIBXSQLDA Methods

The methods of the TIBXSQLDA class are listed here.

Protected Methods

	Name	Description
	GetModified (see page 540)	This is GetModified, a member of class TIBXSQLDA.
	GetNames (see page 541)	This is GetNames, a member of class TIBXSQLDA.
	GetRecordSize (see page 541)	This is GetRecordSize, a member of class TIBXSQLDA.
	GetXSQLDA (see page 541)	This is GetXSQLDA, a member of class TIBXSQLDA.
	GetXSQLVAR (see page 541)	This is GetXSQLVAR, a member of class TIBXSQLDA.
	GetXSQLVARByName (see page 541)	This is GetXSQLVARByName, a member of class TIBXSQLDA.
	Initialize (see page 542)	This is Initialize, a member of class TIBXSQLDA.
	SetCount (see page 542)	This is SetCount, a member of class TIBXSQLDA.

Public Methods

	Name	Description
	AddName (see page 540)	Adds a name to the XSQLDA structure.
	ByName (see page 540)	Returns the XSQLVAR fields by name.

13.1.9.5.1 TIBXSQLDA.AddName Method

Adds a name to the XSQLDA structure.

Pascal

```
procedure AddName(FieldName: String; Idx: Integer);
```

C++

```
__fastcall AddName(UnicodeString FieldName, int Idx);
```

Description

Call AddName to add a name to the XSQLDA structure.

13.1.9.5.2 TIBXSQLDA.ByName Method

Returns the XSQLVAR fields by name.

Pascal

```
function ByName(Idx: String): TIBXSQLVAR;
```

C++

```
__fastcall TIBXSQLVAR ByName(UnicodeString Idx);
```

Description

Use the ByName method to return the XSQLVAR fields by name.

13.1.9.5.3 TIBXSQLDA.GetModified Method

Pascal

```
function GetModified: Boolean;
```

C++

```
__fastcall Boolean GetModified();
```

Description

This is GetModified, a member of class TIBXSQLDA.

13.1.9.5.4 TIBXSQLDA.GetNames Method

Pascal

```
function GetNames: String;
```

C++

```
__fastcall UnicodeString GetNames();
```

Description

This is GetNames, a member of class TIBXSQLDA.

13.1.9.5.5 TIBXSQLDA.GetRecordSize Method

Pascal

```
function GetRecordSize: Integer;
```

C++

```
__fastcall int GetRecordSize();
```

Description

This is GetRecordSize, a member of class TIBXSQLDA.

13.1.9.5.6 TIBXSQLDA.GetXSQLDA Method

Pascal

```
function GetXSQLDA: PXSQLDA;
```

C++

```
__fastcall PXSQLDA GetXSQLDA();
```

Description

This is GetXSQLDA, a member of class TIBXSQLDA.

13.1.9.5.7 TIBXSQLDA.GetXSQLVAR Method

Pascal

```
function GetXSQLVAR(Idx: Integer): TIBXSQLVAR;
```

C++

```
__fastcall TIBXSQLVAR GetXSQLVAR(int Idx);
```

Description

This is GetXSQLVAR, a member of class TIBXSQLDA.

13.1.9.5.8 TIBXSQLDA.GetXSQLVARByName Method

Pascal

```
function GetXSQLVARByName(Idx: String): TIBXSQLVAR;
```

C++

```
__fastcall TIBXSQLVAR GetXSQLVARByName(UnicodeString Idx);
```

Description

This is GetXSQLVARByName, a member of class TIBXSQLDA.

13.1.9.5.9 TIBXSQLDA.Initialize Method

Pascal

```
procedure Initialize;
```

C++

```
__fastcall Initialize();
```

Description

This is Initialize, a member of class TIBXSQLDA.

13.1.9.5.10 TIBXSQLDA.SetCount Method

Pascal

```
procedure SetCount(Value: Integer);
```

C++

```
__fastcall SetCount(int Value);
```

Description

This is SetCount, a member of class TIBXSQLDA.

13.1.9.6 TIBXSQLDA Properties

The properties of the TIBXSQLDA class are listed here.

Public Properties

	Name	Description
R	AsXSQLDA (see page 542)	Represents the XSQLDA field's value as an XSQLDA value.
	Count (see page 543)	Returns the number of XSQLDA fields.
R	Modified (see page 543)	Indicates whether a field has been modified.
R	Names (see page 543)	Returns the XSQLDA field names.
R	RecordSize (see page 543)	Returns the size of the XSQLDA record.
R	UniqueRelationName (see page 544)	Returns the name of the unique relation.
R	Vars (see page 544)	Returns the XSQLVAR defined for the XSQLDA parameter.

13.1.9.6.1 TIBXSQLDA.AsXSQLDA Property

Represents the XSQLDA field's value as an XSQLDA value.

Pascal

```
property AsXSQLDA: PXSQLDA;
```

C++

```
__property PXSQLDA AsXSQLDA;
```

Description

Use the AsXSQLDA property to read the value of the field's data into an object or variable of type XSQLDA, or to assign an

XSQLDA value to the contents of the field.

13.1.9.6.2 TIBXSQLDA.Count Property

Returns the number of XSQLDA fields.

Pascal

```
property Count: Integer;
```

C++

```
__property int Count;
```

Description

Use the Count property to return the number of XSQLDA fields.

13.1.9.6.3 TIBXSQLDA.Modified Property

Indicates whether a field has been modified.

Pascal

```
property Modified: Boolean;
```

C++

```
__property Boolean Modified;
```

Description

Use the Modified property to determine whether a field has been modified.

13.1.9.6.4 TIBXSQLDA.Names Property

Returns the XSQLDA field names.

Pascal

```
property Names: String;
```

C++

```
__property UnicodeString Names;
```

Description

Use the Names property to return the XSQLDA field names.

13.1.9.6.5 TIBXSQLDA.RecordSize Property

Returns the size of the XSQLDA record.

Pascal

```
property RecordSize: Integer;
```

C++

```
__property int RecordSize;
```

Description

Use the RecordSize property to return the XSQLDA record size.

13.1.9.6.6 TIBXSQLDA.UniqueRelationName Property

Returns the name of the unique relation.

Pascal

```
property UniqueRelationName: String;
```

C++

```
__property UnicodeString UniqueRelationName;
```

Description

Use the UniqueRelationName property to return the name of the relation if only one relation is involved in the query. Otherwise, it returns nil (Delphi) or NULL (C++). This property is primarily used for internal purposes.

13.1.9.6.7 TIBXSQLDA.Vars Property

Returns the XSQLVAR defined for the XSQLDA parameter.

Pascal

```
property Vars [Idx: Integer]: TIBXSQLVAR;
```

C++

```
__property TIBXSQLVAR Vars[int Idx];
```

Description

Use Vars to return the XSQLVAR defined for the XSQLDA parameter.

Vars is the default property for TIBXSQLDSA, which means that the name of the Vars property can be omitted when the TIBXSQLDA (see page 536) object implements a property in another object.

13.1.10 TIBXSQLVAR Class

TIBXSQLVAR provides properties and methods for use with the IBSQL (see page 493) component.

Class Hierarchy

Pascal

```
TIBXSQLVAR = class(TObject);
```

C++

```
class TIBXSQLVAR : public TObject;
```

File

IBSQL

Description

Use a TIBXSQLVAR object to provide properties and methods for use with the IBSQL (see page 493) component.

The XSQLVAR structure is a field, sqlvar, in the XSQLDA. The sqlvar is especially important, because one XSQLVAR must be defined for each input parameter or column returned.

Applications do not declare instances of the XSQLVAR ahead of time, but must, instead, dynamically allocate storage for the proper number of XSQLVAR structures required for each DSQL statement before it is executed, then deallocate it, as appropriate, after statement execution.

13.1.10.1 TIBXSQLVAR Members

The following tables list the members exposed by TIBXSQLVAR.

Public Methods

	Name	Description
	Assign (see page 546)	Assigns another XSQLVAR to the XSQLVAR component.
	Clear (see page 546)	Clears the value of this TIBXSQLVAR (see page 544) object.
	Create (see page 545)	Creates an instance of an XSQLVAR structure.
	LoadFromFile (see page 547)	Loads the contents of a file to a Blob field.
	LoadFromStream (see page 547)	Loads a stream into a Blob field.
	SaveToFile (see page 547)	Saves the contents of a Blob field to a file.
	SaveToStream (see page 547)	Saves the contents of a Blob field to a stream.

Public Properties

	Name	Description
	AsCurrency (see page 548)	Represents the XSQLVAR field's value as a Currency value.
	AsDate (see page 548)	Represents the XSQLVAR field's value as a TDateTime value.
	AsDateTime (see page 549)	Represents the XSQLVAR field's value as a TDateTime value.
	AsDouble (see page 549)	Represents the XSQLVAR field's value as a Double value.
	AsFloat (see page 549)	Represents the XSQLVAR field's value as a Float value.
	AsInt64 (see page 550)	Represents the XSQLVAR field's value as a 64-bit integer.
	AsInteger (see page 550)	Represents the XSQLVAR field's value as a 32-bit integer value.
	AsLong (see page 550)	Represents the XSQLVAR field's value as a Long integer value.
	AsPointer (see page 550)	Represents the XSQLVAR field's value as a Pointer value.
	AsQuad (see page 551)	Represents the XSQLVAR field's value as a Quad value.
	AsShort (see page 551)	Represents the XSQLVAR field's value as a Short integer value.
	AsString (see page 551)	Represents the XSQLVAR field's value as a string.
	AsTime (see page 551)	Represents the XSQLVAR field's value as a TDateTime value.
	AsTrimString (see page 552)	Represents the XSQLVAR field's value as a string with any trailing spaces removed.
	AsVariant (see page 552)	Represents the XSQLVAR field's value as type Variant.
	AsXSQLVAR (see page 552)	Represents the XSQLVAR field's value as an XSQLVAR value.
	Data (see page 553)	Accesses the underlying InterBase XSQLVAR structure.
R	Index (see page 553)	Indicates the position of the XSQLVAR in the XSQLDA.
	IsNull (see page 553)	Indicates whether the field has a value assigned to it.
	IsNullable (see page 553)	Indicates whether the field can have a value assigned to it.
	Modified (see page 554)	Indicates whether a field has been modified.
R	Name (see page 554)	Returns the name of the XSQLVAR.
R	Size (see page 554)	Indicates the maximum size, in bytes, of data in the sqldata field of the XSQLVAR.
R	SQLType (see page 554)	Indicates the SQL datatype of parameters or select list items.
	Value (see page 555)	Returns the value of the XSQLVAR field component.

13.1.10.2 TIBXSQLVAR.Create Constructor

Creates an instance of an XSQLVAR structure.

Pascal

```
constructor Create(Parent: TIBXSQLDA; Query: TIBSQL);
```

C++

```
__fastcall TIBXSQLVAR(TIBXSQLDA Parent, TIBSQL Query);
```

Description

Call Create to create an instance of TIBXSQLVAR (see page 544).

Parent is an extended descriptor area that defines the namespace to use for processing the variable name.

Query identifies the query that includes this variable.

13.1.10.3 TIBXSQLVAR Methods

The methods of the TIBXSQLVAR class are listed here.

Public Methods

	Name	Description
	Assign (see page 546)	Assigns another XSQLVAR to the XSQLVAR component.
	Clear (see page 546)	Clears the value of this TIBXSQLVAR (see page 544) object.
	LoadFromFile (see page 547)	Loads the contents of a file to a Blob field.
	LoadFromStream (see page 547)	Loads a stream into a Blob field.
	SaveToFile (see page 547)	Saves the contents of a Blob field to a file.
	SaveToStream (see page 547)	Saves the contents of a Blob field to a stream.

13.1.10.3.1 TIBXSQLVAR.Assign Method

Assigns another XSQLVAR to the XSQLVAR component.

Pascal

```
procedure Assign(Source: TIBXSQLVAR);
```

C++

```
__fastcall Assign(TIBXSQLVAR Source);
```

Description

Use Assign to assign another XSQLVAR to the XSQLVAR component. The fields must have compatible data types.

13.1.10.3.2 TIBXSQLVAR.Clear Method

Clears the value of this TIBXSQLVAR (see page 544) object.

Pascal

```
procedure Clear;
```

C++

```
__fastcall Clear();
```

Description

Call Clear to assign a Null value to this TIBXSQLVAR (see page 544) object.

13.1.10.3.3 TIBXSQLVAR.LoadFromFile Method

Loads the contents of a file to a Blob field.

Pascal

```
procedure LoadFromFile(const FileName: String);
```

C++

```
__fastcall LoadFromFile(const UnicodeString FileName);
```

Description

Call LoadFromFile to load the contents of a file to a Blob field.

13.1.10.3.4 TIBXSQLVAR.LoadFromStream Method

Loads a stream into a Blob field.

Pascal

```
procedure LoadFromStream(Stream: TStream);
```

C++

```
__fastcall LoadFromStream(TStream Stream);
```

Description

Call LoadFromStream to load a stream into a Blob field.

13.1.10.3.5 TIBXSQLVAR.SaveToFile Method

Saves the contents of a Blob field to a file.

Pascal

```
procedure SaveToFile(const FileName: String);
```

C++

```
__fastcall SaveToFile(const UnicodeString FileName);
```

Description

Call SaveToFile to save the contents of a Blob field to a file.

13.1.10.3.6 TIBXSQLVAR.SaveToStream Method

Saves the contents of a Blob field to a stream.

Pascal

```
procedure SaveToStream(Stream: TStream);
```

C++

```
__fastcall SaveToStream(TStream Stream);
```

Description

Call SaveToStream to save the contents of a Blob field to a stream.

13.1.10.4 TIBXSQLVAR Properties

The properties of the TIBXSQLVAR class are listed here.

Public Properties

	Name	Description
	AsCurrency (see page 548)	Represents the XSQLVAR field's value as a Currency value.
	AsDate (see page 548)	Represents the XSQLVAR field's value as a TDateTime value.
	AsDateTime (see page 549)	Represents the XSQLVAR field's value as a TDateTime value.
	AsDouble (see page 549)	Represents the XSQLVAR field's value as a Double value.
	AsFloat (see page 549)	Represents the XSQLVAR field's value as a Float value.
	AsInt64 (see page 550)	Represents the XSQLVAR field's value as a 64-bit integer.
	AsInteger (see page 550)	Represents the XSQLVAR field's value as a 32-bit integer value.
	AsLong (see page 550)	Represents the XSQLVAR field's value as a Long integer value.
	AsPointer (see page 550)	Represents the XSQLVAR field's value as a Pointer value.
	AsQuad (see page 551)	Represents the XSQLVAR field's value as a Quad value.
	AsShort (see page 551)	Represents the XSQLVAR field's value as a Short integer value.
	AsString (see page 551)	Represents the XSQLVAR field's value as a string.
	AsTime (see page 551)	Represents the XSQLVAR field's value as a TDateTime value.
	AsTrimString (see page 552)	Represents the XSQLVAR field's value as a string with any trailing spaces removed.
	AsVariant (see page 552)	Represents the XSQLVAR field's value as type Variant.
	AsXSQLVAR (see page 552)	Represents the XSQLVAR field's value as an XSQLVAR value.
	Data (see page 553)	Accesses the underlying InterBase XSQLVAR structure.
R	Index (see page 553)	Indicates the position of the XSQLVAR in the XSQLDA.
	IsNull (see page 553)	Indicates whether the field has a value assigned to it.
	IsNullable (see page 553)	Indicates whether the field can have a value assigned to it.
	Modified (see page 554)	Indicates whether a field has been modified.
R	Name (see page 554)	Returns the name of the XSQLVAR.
R	Size (see page 554)	Indicates the maximum size, in bytes, of data in the sqldata field of the XSQLVAR.
R	SQLType (see page 554)	Indicates the SQL datatype of parameters or select list items.
	Value (see page 555)	Returns the value of the XSQLVAR field component.

13.1.10.4.1 TIBXSQLVAR.AsCurrency Property

Represents the XSQLVAR field's value as a Currency value.

Pascal

```
property AsCurrency: Currency;
```

C++

```
__property Currency AsCurrency;
```

Description

Use the AsCurrency property to read the value of the XSQLVAR field's data into an object or variable of type Currency, or to assign a Currency value to the contents of the field.

13.1.10.4.2 TIBXSQLVAR.AsDate Property

Represents the XSQLVAR field's value as a TDateTime value.

Pascal

```
property AsDate: TDateTime;
```

C++

```
__property TDateTime AsDate;
```

Description

Use the AsDate property to read the value of the field's data into an object or variable of type TDateTime, or to assign a TDateTime value to the contents of the field. The Time portion of the TDateTime value is set to zero.

13.1.10.4.3 TIBXSQLVAR.AsDateTime Property

Represents the XSQLVAR field's value as a TDateTime value.

Pascal

```
property AsDateTime: TDateTime;
```

C++

```
__property TDateTime AsDateTime;
```

Description

Use the AsDateTime property to read the value of the field's data into an object or variable of type TDateTime, or to assign a TDateTime value to the contents of the field.

13.1.10.4.4 TIBXSQLVAR.AsDouble Property

Represents the XSQLVAR field's value as a Double value.

Pascal

```
property AsDouble: Double;
```

C++

```
__property Double AsDouble;
```

Description

Use the AsDouble property to read the value of the field's data into an object or variable of type Double, or to assign a Double value to the contents of the field.

13.1.10.4.5 TIBXSQLVAR.AsFloat Property

Represents the XSQLVAR field's value as a Float value.

Pascal

```
property AsFloat: Float;
```

C++

```
__property Float AsFloat;
```

Description

Use the AsDouble (see page 549) property to read the value of the field's data into an object or variable of type Float, or to assign a Float value to the contents of the field.

13.1.10.4.6 TIBXSQLVAR.AsInt64 Property

Represents the XSQLVAR field's value as a 64-bit integer.

Pascal

```
property AsInt64: Int64;
```

C++

```
__property Int64 AsInt64;
```

Description

Use the AsInt64 property to read the value of the field's data into a 64-bit integer, or to assign an 64-bit integer value to the contents of the field.

13.1.10.4.7 TIBXSQLVAR.AsInteger Property

Represents the XSQLVAR field's value as a 32-bit integer value.

Pascal

```
property AsInteger: Integer;
```

C++

```
__property int AsInteger;
```

Description

Use the AsInteger property to read the value of the field's data into an integer, or to assign an integer value to the contents of the field.

13.1.10.4.8 TIBXSQLVAR.AsLong Property

Represents the XSQLVAR field's value as a Long integer value.

Pascal

```
property AsLong: Long;
```

C++

```
__property Long AsLong;
```

Description

Use the AsLong property to read the value of the field's data into a Long integer, or to assign a Long integer value to the contents of the field.

13.1.10.4.9 TIBXSQLVAR.AsPointer Property

Represents the XSQLVAR field's value as a Pointer value.

Pascal

```
property AsPointer: Pointer;
```

C++

```
__property void * AsPointer;
```

Description

Use the AsPointer property to read the value of the field's data into an object or variable of type Pointer, or to assign a Pointer value to the contents of the field.

13.1.10.4.10 TIBXSQLVAR.AsQuad Property

Represents the XSQLVAR field's value as a Quad value.

Pascal

```
property AsQuad: TISC_QUAD;
```

C++

```
__property TISC_QUAD AsQuad;
```

Description

Use the AsQuad property to read the value of the field's data into an object or variable of type Quad, or to assign a Quad value to the contents of the field.

13.1.10.4.11 TIBXSQLVAR.AsShort Property

Represents the XSQLVAR field's value as a Short integer value.

Pascal

```
property AsShort: Short;
```

C++

```
__property Short AsShort;
```

Description

Use the AsShort property to to read the value of the field's data into a Short integer, or to assign a Short integer value to the contents of the field.

13.1.10.4.12 TIBXSQLVAR.AsString Property

Represents the XSQLVAR field's value as a string.

Pascal

```
property AsString: String;
```

C++

```
__property UnicodeString AsString;
```

Description

Use the AsString property to read the value of the field's data into a string, or to assign a String value to the contents of the field.

13.1.10.4.13 TIBXSQLVAR.AsTime Property

Represents the XSQLVAR field's value as a TDateTime value.

Pascal

```
property AsTime: TDateTime;
```

C++

```
__property TDateTime AsTime;
```

Description

Use the AsTime property to read the value of the field's data into an object or variable of type TDateTime, or to assign a TDateTime value to the contents of the field. The Date portion of the field is set to zero.

13.1.10.4.14 TIBXSQLVAR.AsTrimString Property

Represents the XSQLVAR field's value as a string with any trailing spaces removed.

Pascal

```
property AsTrimString: String;
```

C++

```
__property UnicodeString AsTrimString;
```

Description

Use the AsTrimString property to read the value of the field's data into a string, removing any trailing spaces from the value.

When assigning AsTrimString, any trailing spaces in the assigned string are removed before assigning the value. The spaces do not become part of the underlying value.

13.1.10.4.15 TIBXSQLVAR.AsVariant Property

Represents the XSQLVAR field's value as type Variant.

Pascal

```
property AsVariant: Variant;
```

C++

```
__property Variant AsVariant;
```

Description

Use the AsVariant property to read the value of the field's data into a Variant, or to assign a Variant value to the contents of the field.

13.1.10.4.16 TIBXSQLVAR.AsXSQLVAR Property

Represents the XSQLVAR field's value as an XSQLVAR value.

Pascal

```
property AsXSQLVAR: PXSQLVAR;
```

C++

```
__property PXSQLVAR AsXSQLVAR;
```

Description

Use the AsXSQLVAR property to read the value of the field's data into an object or variable of type XSQLVAR, or to assign an XSQLVAR value to the contents of the field.

13.1.10.4.17 TIBXSQLVAR.Data Property

Accesses the underlying InterBase XSQLVAR structure.

Pascal

```
property Data: PXSQLVAR;
```

C++

```
__property PXSQLVAR Data;
```

Description

Use the Data property to access the underlying InterBase XSQLVAR structure.

13.1.10.4.18 TIBXSQLVAR.Index Property

Indicates the position of the XSQLVAR in the XSQLDA.

Pascal

```
property Index: Integer;
```

C++

```
__property int Index;
```

Description

Use the Index property to obtain the position of the XSQLVAR in the XSQLDA.

13.1.10.4.19 TIBXSQLVAR.IsNull Property

Indicates whether the field has a value assigned to it.

Pascal

```
property IsNull: Boolean;
```

C++

```
__property Boolean IsNull;
```

Description

Use the IsNull property to determine if the field contains a value. If IsNull is true, the field is blank. If IsNull is false, the field has a value.

13.1.10.4.20 TIBXSQLVAR.IsNullable Property

Indicates whether the field can have a value assigned to it.

Pascal

```
property IsNullable: Boolean;
```

C++

```
__property Boolean IsNullable;
```

Description

Use the IsNullable property to determine if the field can contain a value. If IsNullable is true, the field can contain a value. If IsNullable is false, the field cannot contain a value.

13.1.10.4.21 TIBXSQLVAR.Modified Property

Indicates whether a field has been modified.

Pascal

```
property Modified: Boolean;
```

C++

```
__property Boolean Modified;
```

Description

Use the Modified property to determine whether a field has been modified.

13.1.10.4.22 TIBXSQLVAR.Name Property

Returns the name of the XSQLVAR.

Pascal

```
property Name: String;
```

C++

```
__property UnicodeString Name;
```

Description

Use the Name property to return the name of the XSQLVAR.

13.1.10.4.23 TIBXSQLVAR.Size Property

Indicates the maximum size, in bytes, of data in the sqldata field of the XSQLVAR.

Pascal

```
property Size: Integer;
```

C++

```
__property int Size;
```

Description

Use the Size property to return the maximum size, in bytes, of data in the sqldata field of the XSQLVAR.

13.1.10.4.24 TIBXSQLVAR.SQLType Property

Indicates the SQL datatype of parameters or select list items.

Pascal

```
property SQLType: Integer;
```

C++

```
__property int SQLType;
```

Description

Read the SQLType property to indicate the SQL datatype of parameters or select list items.

13.1.10.4.25 TIBXSQLVAR.Value Property

Returns the value of the XSQLVAR field component.

Pascal

```
property Value: Variant;
```

C++

```
__property Variant Value;
```

Description

Use Value to return the value of the XSQLVAR field component as a Variant.

13.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TIBSQLTypes (see page 556)	TIBSQLTypes indicates the type of an SQL query against an InterBase database.

13.2.1 IBSQL.TIBSQLTypes Enumeration

TIBSQLTypes indicates the type of an SQL query against an InterBase database.

Pascal

```
TIBSQLTypes = (
  SQLUnknown,
  SQLSelect,
  SQLInsert,
  SQLUpdate,
  SQLDelete,
  SQLDDL,
  SQLGetSegment,
  SQLPutSegment,
  SQLExecProcedure,
  SQLStartTransaction,
  SQLCommit,
  SQLRollback,
  SQLSelectForUpdate,
  SQLSetGenerator
);
```

C++

```
enum TIBSQLTypes {
  SQLUnknown,
  SQLSelect,
  SQLInsert,
  SQLUpdate,
  SQLDelete,
  SQLDDL,
  SQLGetSegment,
  SQLPutSegment,
  SQLExecProcedure,
  SQLStartTransaction,
  SQLCommit,
  SQLRollback,
  SQLSelectForUpdate,
  SQLSetGenerator
};
```

File

IBSQL

Description

TIBSQLTypes values are:

SQLCommit	Commits an active transaction
SQLDDL	Executes a DDL statement
SQLDelete	Removes rows in a table or in the active set of a cursor
SQLExecProcedure	Calls a stored procedure
SQLGetSegment	Reads a segment from an open Blob
SQLInsert	Adds one or more new rows to a specified table
SQLPutSegment	Writes a Blob segment
SQLRollback	Restores the database to its state prior to the start of the current transaction
SQLSetForUpdate	Stored procedure is set for updating
SQLSetGenerator	Sets a new value for an existing generator
SQLSelect	Retrieves data from one or more tables
SQLStartTransaction	Starts a new transaction against one or more databases
SQLUnknown	Unknown SQL type
SQLUpdate	Changes data in all or part of an existing row in a table, view, or active set of a cursor

13.3 Types

The following table lists types in this documentation.

Types

Name	Description
TIBXSQLVARArray (see page 558)	This is type IBSQL.TIBXSQLVARArray.

13.3.1 IBSQL.TIBXSQLVARArray Type

Pascal

```
TIBXSQLVARArray = Array of TIBXSQLVAR;
```

C++

```
Array of TIBXSQLVAR TIBXSQLVARArray;
```

File

IBSQL

Description

This is type IBSQL.TIBXSQLVARArray.

14 IBSQLMonitor Namespace

This is namespace IBSQLMonitor.

Topics

Name	Description
Classes (see page 561)	The following table lists classes in this documentation.
Interfaces (see page 567)	The following table lists interfaces in this documentation.
Functions (see page 575)	The following table lists functions in this documentation.
Types (see page 577)	The following table lists types in this documentation.
Constants (see page 578)	The following table lists constants in this documentation.

Classes

	Name	Description
	TIBCustomSQLMonitor (see page 561)	TIBCustomSQLMonitor is the ancestor for TIBSQLMonitor (see page 564).
	TIBSQLMonitor (see page 564)	TIBSQLMonitor monitors dynamic SQL passed to the InterBase server.

Constants

Name	Description
WM_IBSQL_SQL_EVENT (see page 578)	This is constant IBSQLMonitor.WM_IBSQL_SQL_EVENT.
WM_MAX_IBSQL_MONITOR (see page 578)	This is constant IBSQLMonitor.WM_MAX_IBSQL_MONITOR.
WM_MIN_IBSQL_MONITOR (see page 578)	This is constant IBSQLMonitor.WM_MIN_IBSQL_MONITOR.

Functions

	Name	Description
	DisableMonitoring (see page 575)	Disables SQL monitoring.
	EnableMonitoring (see page 575)	This is function IBSQLMonitor.EnableMonitoring.
	MonitorHook (see page 576)	This is function IBSQLMonitor.MonitorHook.
	MonitoringEnabled (see page 576)	This is function IBSQLMonitor.MonitoringEnabled.

Interfaces

	Name	Description
	IIBSQLMonitorHook (see page 567)	This is class IBSQLMonitor.IIBSQLMonitorHook.

Types

Name	Description
TSQLEvent (see page 577)	TSQLEvent is the type of event handlers that respond to activity on an InterBase server.

14.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBCustomSQLMonitor (see page 561)	TIBCustomSQLMonitor is the ancestor for TIBSQLMonitor (see page 564).
	TIBSQLMonitor (see page 564)	TIBSQLMonitor monitors dynamic SQL passed to the InterBase server.

14.1.1 TIBCustomSQLMonitor Class

TIBCustomSQLMonitor is the ancestor for TIBSQLMonitor (see page 564).

Class Hierarchy

Pascal

```
TIBCustomSQLMonitor = class(TComponent);
```

C++

```
class TIBCustomSQLMonitor : public TComponent;
```

File

IBSQLMonitor

Description

TIBCustomSQLMonitor is the ancestor object from which TIBSQLMonitor (see page 564) is derived.

14.1.1.1 TIBCustomSQLMonitor Members

The following tables list the members exposed by TIBCustomSQLMonitor.

Protected Events

	Name	Description
	OnSQL (see page 564)	Reports dynamic SQL activity on InterBase applications.

Protected Properties

	Name	Description
	Enabled (see page 563)	Indicates whether the IB (see page 1) SQL monitor is actively monitoring communication with the server.
	TraceFlags (see page 564)	Indicates which database operations are tracked.

Public Methods

	Name	Description
	Create (see page 562)	Creates an instance of TIBCustomSQLMonitor (see page 561).
	Destroy (see page 562)	Destroys the instance of TIBCustomSQLMonitor (see page 561).
	Release (see page 562)	Frees the helper objects that intercept database commands.

Public Properties

	Name	Description
	Handle (see page 563)	Specifies the window handle that the SQL monitor uses to receive asynchronous messages.

14.1.1.2 TIBCustomSQLMonitor.Create Constructor

Creates an instance of TIBCustomSQLMonitor (see page 561).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBCustomSQLMonitor(TComponent * AOwner);
```

Description

Call Create to create an instance of TIBCustomSQLMonitor (see page 561). Create also:

1. Creates the window
2. Registers the event
3. Fires the Monitor thread

14.1.1.3 TIBCustomSQLMonitor.Destroy Destructor

Destroys the instance of TIBCustomSQLMonitor (see page 561).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBCustomSQLMonitor();
```

Description

Do not call Destroy directly in an application. Instead, an application should call Free. Free verifies that the service object has not already been freed before it calls Destroy.

Destroy unregisters the event, destroys the window, and kills the Monitor thread.

14.1.1.4 TIBCustomSQLMonitor Methods

The methods of the TIBCustomSQLMonitor class are listed here.

Public Methods

	Name	Description
	Release (see page 562)	Frees the helper objects that intercept database commands.

14.1.1.4.1 TIBCustomSQLMonitor.Release Method

Frees the helper objects that intercept database commands.

Pascal

```
procedure Release;
```

C++

```
__fastcall Release();
```

Description

Call Release when the SQL monitor will not be monitoring any more messages. Release sends a message to the object that intercepts database commands, telling it to free itself.

To temporarily disable the monitor, use the Enabled (see page 563) property instead.

14.1.1.5 TIBCustomSQLMonitor Properties

The properties of the TIBCustomSQLMonitor class are listed here.

Protected Properties

	Name	Description
	Enabled (see page 563)	Indicates whether the IB (see page 1) SQL monitor is actively monitoring communication with the server.
	TraceFlags (see page 564)	Indicates which database operations are tracked.

Public Properties

	Name	Description
R	Handle (see page 563)	Specifies the window handle that the SQL monitor uses to receive asynchronous messages.

14.1.1.5.1 TIBCustomSQLMonitor.Enabled Property

Indicates whether the IB (see page 1) SQL monitor is actively monitoring communication with the server.

Pascal

```
property Enabled: Boolean;
```

C++

```
__property Boolean Enabled;
```

Description

Use Enabled to turn the SQL monitor on and off. When Enabled is true, the SQL monitor receives an OnSQL (see page 564) event for every database operation specified by the TraceFlags (see page 564) property. When Enabled is false, the SQL monitor does not receive any events.

14.1.1.5.2 TIBCustomSQLMonitor.Handle Property

Specifies the window handle that the SQL monitor uses to receive asynchronous messages.

Pascal

```
property Handle: HWND;
```

C++

```
__property HWND Handle;
```

Description

A separate thread of execution monitors the database operations that trigger OnSQL (see page 564) events. When that thread

detects a relevant database operation, it sends a Windows message to the window identified by this Handle.

14.1.1.5.3 TIBCustomSQLMonitor.TraceFlags Property

Indicates which database operations are tracked.

Pascal

```
property TraceFlags: TTraceFlags;
```

C++

```
__property TTraceFlags TraceFlags;
```

Description

Use TraceFlags to specify which database operations the SQL Monitor should track in an application at runtime. TraceFlags enables performance tuning and SQL debugging when working with remote SQL database servers.

Note: Normally trace options are set from the SQL Monitor rather than setting TraceFlags in application code.

14.1.1.6 TIBCustomSQLMonitor Events

The events of the TIBCustomSQLMonitor class are listed here.

Protected Events

	Name	Description
	OnSQL (see page 564)	Reports dynamic SQL activity on InterBase applications.

14.1.1.6.1 TIBCustomSQLMonitor.OnSQL Event

Reports dynamic SQL activity on InterBase applications.

Pascal

```
property OnSQL: TSQLEvent;
```

C++

```
__property TSQLEvent OnSQL;
```

Description

Write an OnSQL event handler to report dynamic SQL activity on InterBase applications.

Note: You must enable the TraceFlags (see page 564) in each TIBDatabase component in order for the SQL monitor to receive status information from each database connection.

14.1.2 TIBSQLMonitor Class

TIBSQLMonitor monitors dynamic SQL passed to the InterBase server.

Class Hierarchy

Pascal

```
TIBSQLMonitor = class(TIBCustomSQLMonitor);
```

C++

```
class TIBSQLMonitor : public TIBCustomSQLMonitor;
```

File

IBSQLMonitor

Description

Use TIBSQLMonitor to monitor dynamic SQL taking place in InterBase applications. Enable TraceFlags (see page 566) in each TIBDatabase component in order for the SQL monitor to receive status information from each database connection.

See Also

Getting Started with InterBase Express

14.1.2.1 TIBSQLMonitor Members

The following tables list the members exposed by TIBSQLMonitor.

Public Methods

	Name	Description
	Create (see page 562)	Creates an instance of TIBCustomSQLMonitor (see page 561).
	Destroy (see page 562)	Destroys the instance of TIBCustomSQLMonitor (see page 561).
	Release (see page 562)	Frees the helper objects that intercept database commands.

Public Properties

	Name	Description
	Handle (see page 563)	Specifies the window handle that the SQL monitor uses to receive asynchronous messages.

Published Events

	Name	Description
	OnSQL (see page 566)	Reports dynamic SQL activity on InterBase applications.

Published Properties

	Name	Description
	Enabled (see page 565)	Indicates whether the IB SQL monitor is actively monitoring communication with the server.
	TraceFlags (see page 566)	Indicates which database operations are tracked.

14.1.2.2 TIBSQLMonitor Properties

The properties of the TIBSQLMonitor class are listed here.

Published Properties

	Name	Description
	Enabled (see page 565)	Indicates whether the IB SQL monitor is actively monitoring communication with the server.
	TraceFlags (see page 566)	Indicates which database operations are tracked.

14.1.2.2.1 TIBSQLMonitor.Enabled Property

Indicates whether the IB SQL monitor is actively monitoring communication with the server.

Pascal

```
property Enabled: Boolean;
```

C++

```
__property Boolean Enabled;
```

Description

Use Enabled to turn the SQL monitor on and off. When Enabled is true, the SQL monitor receives an OnSQL event for every database operation specified by the TraceFlags property. When Enabled is false, the SQL monitor does not receive any events.

14.1.2.2.2 TIBSQLMonitor.TraceFlags Property

Indicates which database operations are tracked.

Pascal

```
property TraceFlags: TTraceFlags;
```

C++

```
__property TTraceFlags TraceFlags;
```

Description

Use TraceFlags to specify which database operations the SQL Monitor should track in an application at runtime. TraceFlags enables performance tuning and SQL debugging when working with remote SQL database servers.

Note: Normally trace options are set from the SQL Monitor rather than setting TraceFlags in application code.

14.1.2.3 TIBSQLMonitor Events

The events of the TIBSQLMonitor class are listed here.

Published Events

	Name	Description
	OnSQL (see page 566)	Reports dynamic SQL activity on InterBase applications.

14.1.2.3.1 TIBSQLMonitor.OnSQL Event

Reports dynamic SQL activity on InterBase applications.

Pascal

```
property OnSQL: TSQLEvent;
```

C++

```
__property TSQLEvent OnSQL;
```

Description

Write an OnSQL event handler to report dynamic SQL activity on InterBase applications.

Note: You must enable the TraceFlags in each TIBDatabase component in order for the SQL monitor to receive status information from each database connection.

14.2 Interfaces

The following table lists interfaces in this documentation.

Interfaces

	Name	Description
	IIBSQLMonitorHook (see page 567)	This is class IBSQLMonitor.IIBSQLMonitorHook.

14.2.1 IIBSQLMonitorHook Interface

Class Hierarchy

IBSQLMonitor.IIBSQLMonitorHook

Pascal

```
IIBSQLMonitorHook = interface;
```

C++

```
__interface IIBSQLMonitorHook;
```

File

IBSQLMonitor

Description

This is class IBSQLMonitor.IIBSQLMonitorHook.

14.2.1.1 IIBSQLMonitorHook Members

The following tables list the members exposed by IIBSQLMonitorHook.

Public Methods

	Name	Description
	DBConnect (see page 568)	This is DBConnect, a member of class IIBSQLMonitorHook.
	DBDisconnect (see page 569)	This is DBDisconnect, a member of class IIBSQLMonitorHook.
	GetEnabled (see page 569)	This is GetEnabled, a member of class IIBSQLMonitorHook.
	GetMonitorCount (see page 569)	This is GetMonitorCount, a member of class IIBSQLMonitorHook.
	GetTraceFlags (see page 569)	This is GetTraceFlags, a member of class IIBSQLMonitorHook.
	RegisterMonitor (see page 570)	This is RegisterMonitor, a member of class IIBSQLMonitorHook.
	ReleaseMonitor (see page 570)	This is ReleaseMonitor, a member of class IIBSQLMonitorHook.
	SendMisc (see page 570)	This is SendMisc, a member of class IIBSQLMonitorHook.
	ServiceAttach (see page 570)	This is ServiceAttach, a member of class IIBSQLMonitorHook.
	ServiceDetach (see page 570)	This is ServiceDetach, a member of class IIBSQLMonitorHook.
	ServiceQuery (see page 571)	This is ServiceQuery, a member of class IIBSQLMonitorHook.
	ServiceStart (see page 571)	This is ServiceStart, a member of class IIBSQLMonitorHook.
	SetEnabled (see page 571)	This is SetEnabled, a member of class IIBSQLMonitorHook.
	SetTraceFlags (see page 571)	This is SetTraceFlags, a member of class IIBSQLMonitorHook.
	SQLExecute (see page 572)	This is SQLExecute, a member of class IIBSQLMonitorHook.

	SQLFetch (see page 572)	This is SQLFetch, a member of class IIBSQLMonitorHook.
	SQLPrepare (see page 572)	This is SQLPrepare, a member of class IIBSQLMonitorHook.
	TRCommit (see page 572)	This is TRCommit, a member of class IIBSQLMonitorHook.
	TRCommitRetaining (see page 572)	This is TRCommitRetaining, a member of class IIBSQLMonitorHook.
	TRRollback (see page 573)	This is TRRollback, a member of class IIBSQLMonitorHook.
	TRRollbackRetaining (see page 573)	This is TRRollbackRetaining, a member of class IIBSQLMonitorHook.
	TRStart (see page 573)	This is TRStart, a member of class IIBSQLMonitorHook.
	UnregisterMonitor (see page 573)	This is UnregisterMonitor, a member of class IIBSQLMonitorHook.

Public Properties

	Name	Description
	Enabled (see page 574)	This is Enabled, a member of class IIBSQLMonitorHook.
	TraceFlags (see page 574)	This is TraceFlags, a member of class IIBSQLMonitorHook.

14.2.1.2 IIBSQLMonitorHook Methods

The methods of the IIBSQLMonitorHook class are listed here.

Public Methods

	Name	Description
	DBConnect (see page 568)	This is DBConnect, a member of class IIBSQLMonitorHook.
	DBDisconnect (see page 569)	This is DBDisconnect, a member of class IIBSQLMonitorHook.
	GetEnabled (see page 569)	This is GetEnabled, a member of class IIBSQLMonitorHook.
	GetMonitorCount (see page 569)	This is GetMonitorCount, a member of class IIBSQLMonitorHook.
	GetTraceFlags (see page 569)	This is GetTraceFlags, a member of class IIBSQLMonitorHook.
	RegisterMonitor (see page 570)	This is RegisterMonitor, a member of class IIBSQLMonitorHook.
	ReleaseMonitor (see page 570)	This is ReleaseMonitor, a member of class IIBSQLMonitorHook.
	SendMisc (see page 570)	This is SendMisc, a member of class IIBSQLMonitorHook.
	ServiceAttach (see page 570)	This is ServiceAttach, a member of class IIBSQLMonitorHook.
	ServiceDetach (see page 570)	This is ServiceDetach, a member of class IIBSQLMonitorHook.
	ServiceQuery (see page 571)	This is ServiceQuery, a member of class IIBSQLMonitorHook.
	ServiceStart (see page 571)	This is ServiceStart, a member of class IIBSQLMonitorHook.
	SetEnabled (see page 571)	This is SetEnabled, a member of class IIBSQLMonitorHook.
	SetTraceFlags (see page 571)	This is SetTraceFlags, a member of class IIBSQLMonitorHook.
	SQLExecute (see page 572)	This is SQLExecute, a member of class IIBSQLMonitorHook.
	SQLFetch (see page 572)	This is SQLFetch, a member of class IIBSQLMonitorHook.
	SQLPrepare (see page 572)	This is SQLPrepare, a member of class IIBSQLMonitorHook.
	TRCommit (see page 572)	This is TRCommit, a member of class IIBSQLMonitorHook.
	TRCommitRetaining (see page 572)	This is TRCommitRetaining, a member of class IIBSQLMonitorHook.
	TRRollback (see page 573)	This is TRRollback, a member of class IIBSQLMonitorHook.
	TRRollbackRetaining (see page 573)	This is TRRollbackRetaining, a member of class IIBSQLMonitorHook.
	TRStart (see page 573)	This is TRStart, a member of class IIBSQLMonitorHook.
	UnregisterMonitor (see page 573)	This is UnregisterMonitor, a member of class IIBSQLMonitorHook.

14.2.1.2.1 IIBSQLMonitorHook.DBConnect Method

Pascal

```
procedure DBConnect(db: TIBDatabase);
```

C++

```
__fastcall DBConnect(TIBDatabase db);
```

Description

This is DBConnect, a member of class IIBSQLMonitorHook.

14.2.1.2.2 IIBSQLMonitorHook.DBDisconnect Method

Pascal

```
procedure DBDisconnect(db: TIBDatabase);
```

C++

```
__fastcall DBDisconnect(TIBDatabase db);
```

Description

This is DBDisconnect, a member of class IIBSQLMonitorHook.

14.2.1.2.3 IIBSQLMonitorHook.GetEnabled Method

Pascal

```
function GetEnabled: boolean;
```

C++

```
__fastcall boolean GetEnabled();
```

Description

This is GetEnabled, a member of class IIBSQLMonitorHook.

14.2.1.2.4 IIBSQLMonitorHook.GetMonitorCount Method

Pascal

```
function GetMonitorCount: Integer;
```

C++

```
__fastcall int GetMonitorCount();
```

Description

This is GetMonitorCount, a member of class IIBSQLMonitorHook.

14.2.1.2.5 IIBSQLMonitorHook.GetTraceFlags Method

Pascal

```
function GetTraceFlags: TTraceFlags;
```

C++

```
__fastcall TTraceFlags GetTraceFlags();
```

Description

This is GetTraceFlags, a member of class IIBSQLMonitorHook.

14.2.1.2.6 IIBSQLMonitorHook.RegisterMonitor Method

Pascal

```
procedure RegisterMonitor(SQLMonitor: TIBCustomSQLMonitor);
```

C++

```
__fastcall RegisterMonitor(TIBCustomSQLMonitor SQLMonitor);
```

Description

This is RegisterMonitor, a member of class IIBSQLMonitorHook.

14.2.1.2.7 IIBSQLMonitorHook.ReleaseMonitor Method

Pascal

```
procedure ReleaseMonitor(Arg: TIBCustomSQLMonitor);
```

C++

```
__fastcall ReleaseMonitor(TIBCustomSQLMonitor Arg);
```

Description

This is ReleaseMonitor, a member of class IIBSQLMonitorHook.

14.2.1.2.8 IIBSQLMonitorHook.SendMisc Method

Pascal

```
procedure SendMisc(Msg: String);
```

C++

```
__fastcall SendMisc(UnicodeString Msg);
```

Description

This is SendMisc, a member of class IIBSQLMonitorHook.

14.2.1.2.9 IIBSQLMonitorHook.ServiceAttach Method

Pascal

```
procedure ServiceAttach(service: TIBCustomService);
```

C++

```
__fastcall ServiceAttach(TIBCustomService service);
```

Description

This is ServiceAttach, a member of class IIBSQLMonitorHook.

14.2.1.2.10 IIBSQLMonitorHook.ServiceDetach Method

Pascal

```
procedure ServiceDetach(service: TIBCustomService);
```

C++

```
__fastcall ServiceDetach(TIBCustomService service);
```

Description

This is ServiceDetach, a member of class IIBSQLMonitorHook.

14.2.1.2.11 IIBSQLMonitorHook.ServiceQuery Method

Pascal

```
procedure ServiceQuery(service: TIBCustomService);
```

C++

```
__fastcall ServiceQuery(TIBCustomService service);
```

Description

This is ServiceQuery, a member of class IIBSQLMonitorHook.

14.2.1.2.12 IIBSQLMonitorHook.ServiceStart Method

Pascal

```
procedure ServiceStart(service: TIBCustomService);
```

C++

```
__fastcall ServiceStart(TIBCustomService service);
```

Description

This is ServiceStart, a member of class IIBSQLMonitorHook.

14.2.1.2.13 IIBSQLMonitorHook.SetEnabled Method

Pascal

```
procedure SetEnabled(const Value: Boolean);
```

C++

```
__fastcall SetEnabled(const Boolean Value);
```

Description

This is SetEnabled, a member of class IIBSQLMonitorHook.

14.2.1.2.14 IIBSQLMonitorHook.SetTraceFlags Method

Pascal

```
procedure SetTraceFlags(const Value: TTraceFlags);
```

C++

```
__fastcall SetTraceFlags(const TTraceFlags Value);
```

Description

This is SetTraceFlags, a member of class IIBSQLMonitorHook.

14.2.1.2.15 IIBSQLMonitorHook.SQLExecute Method

Pascal

```
procedure SQLExecute(qry: TIBSQL);
```

C++

```
__fastcall SQLExecute(TIBSQL qry);
```

Description

This is SQLExecute, a member of class IIBSQLMonitorHook.

14.2.1.2.16 IIBSQLMonitorHook.SQLFetch Method

Pascal

```
procedure SQLFetch(qry: TIBSQL);
```

C++

```
__fastcall SQLFetch(TIBSQL qry);
```

Description

This is SQLFetch, a member of class IIBSQLMonitorHook.

14.2.1.2.17 IIBSQLMonitorHook.SQLPrepare Method

Pascal

```
procedure SQLPrepare(qry: TIBSQL);
```

C++

```
__fastcall SQLPrepare(TIBSQL qry);
```

Description

This is SQLPrepare, a member of class IIBSQLMonitorHook.

14.2.1.2.18 IIBSQLMonitorHook.TRCommit Method

Pascal

```
procedure TRCommit(tr: TIBTransaction);
```

C++

```
__fastcall TRCommit(TIBTransaction tr);
```

Description

This is TRCommit, a member of class IIBSQLMonitorHook.

14.2.1.2.19 IIBSQLMonitorHook.TRCommitRetaining Method

Pascal

```
procedure TRCommitRetaining(tr: TIBTransaction);
```

C++

```
__fastcall TRCommitRetaining(TIBTransaction tr);
```

Description

This is TRCommitRetaining, a member of class IIBSQLMonitorHook.

14.2.1.2.20 IIBSQLMonitorHook.TRRollback Method

Pascal

```
procedure TRRollback(tr: TIBTransaction);
```

C++

```
__fastcall TRRollback(TIBTransaction tr);
```

Description

This is TRRollback, a member of class IIBSQLMonitorHook.

14.2.1.2.21 IIBSQLMonitorHook.TRRollbackRetaining Method

Pascal

```
procedure TRRollbackRetaining(tr: TIBTransaction);
```

C++

```
__fastcall TRRollbackRetaining(TIBTransaction tr);
```

Description

This is TRRollbackRetaining, a member of class IIBSQLMonitorHook.

14.2.1.2.22 IIBSQLMonitorHook.TRStart Method

Pascal

```
procedure TRStart(tr: TIBTransaction);
```

C++

```
__fastcall TRStart(TIBTransaction tr);
```

Description

This is TRStart, a member of class IIBSQLMonitorHook.

14.2.1.2.23 IIBSQLMonitorHook.UnregisterMonitor Method

Pascal

```
procedure UnregisterMonitor(SQLMonitor: TIBCustomSQLMonitor);
```

C++

```
__fastcall UnregisterMonitor(TIBCustomSQLMonitor SQLMonitor);
```

Description

This is UnregisterMonitor, a member of class IIBSQLMonitorHook.

14.2.1.3 IIBSQLMonitorHook Properties

The properties of the IIBSQLMonitorHook class are listed here.

Public Properties

	Name	Description
	Enabled (see page 574)	This is Enabled, a member of class IIBSQLMonitorHook.
	TraceFlags (see page 574)	This is TraceFlags, a member of class IIBSQLMonitorHook.

14.2.1.3.1 IIBSQLMonitorHook.Enabled Property

Pascal

```
property Enabled: Boolean;
```

C++

```
__property Boolean Enabled;
```

Description

This is Enabled, a member of class IIBSQLMonitorHook.

14.2.1.3.2 IIBSQLMonitorHook.TraceFlags Property

Pascal

```
property TraceFlags: TTraceFlags;
```

C++

```
__property TTraceFlags TraceFlags;
```

Description

This is TraceFlags, a member of class IIBSQLMonitorHook.

14.3 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	DisableMonitoring (see page 575)	Disables SQL monitoring.
	EnableMonitoring (see page 575)	This is function IBSQLMonitor.EnableMonitoring.
	MonitorHook (see page 576)	This is function IBSQLMonitor.MonitorHook.
	MonitoringEnabled (see page 576)	This is function IBSQLMonitor.MonitoringEnabled.

14.3.1 IBSQLMonitor.DisableMonitoring Function

Disables SQL monitoring.

Pascal

```
procedure DisableMonitoring;
```

C++

```
DisableMonitoring();
```

File

IBSQLMonitor

Description

Use the DisableMonitoring procedure to disable SQL monitoring.

14.3.2 IBSQLMonitor.EnableMonitoring Function

Pascal

```
procedure EnableMonitoring;
```

C++

```
EnableMonitoring();
```

File

IBSQLMonitor

Description

This is function IBSQLMonitor.EnableMonitoring.

14.3.3 IBSQLMonitor.MonitorHook Function

Pascal

```
function MonitorHook: IIBSQLMonitorHook;
```

C++

```
IIBSQLMonitorHook MonitorHook();
```

File

IBSQLMonitor

Description

This is function IBSQLMonitor.MonitorHook.

14.3.4 IBSQLMonitor.MonitoringEnabled Function

Pascal

```
function MonitoringEnabled: Boolean;
```

C++

```
Boolean MonitoringEnabled();
```

File

IBSQLMonitor

Description

This is function IBSQLMonitor.MonitoringEnabled.

14.4 Types

The following table lists types in this documentation.

Types

Name	Description
TSQLEvent (see page 577)	TSQLEvent is the type of event handlers that respond to activity on an InterBase server.

14.4.1 IBSQLMonitor.TSQLEvent Type

TSQLEvent is the type of event handlers that respond to activity on an InterBase server.

Pascal

```
TSQLEvent = procedure (EventText: String; EventTime : TDateTime) of object;
```

C++

```
(EventText: String; EventTime : TDateTime) ( TSQLEvent)();
```

File

IBSQLMonitor

Description

EventText is the text of the SQL command that triggered the event.

EventTime indicates the time when the command passed to or from the InterBase server.

See Also

Procedural types

Creating events

14.5 Constants

The following table lists constants in this documentation.

Constants

Name	Description
WM_IBSQL_SQL_EVENT (see page 578)	This is constant IBSQLMonitor.WM_IBSQL_SQL_EVENT.
WM_MAX_IBSQL_MONITOR (see page 578)	This is constant IBSQLMonitor.WM_MAX_IBSQL_MONITOR.
WM_MIN_IBSQL_MONITOR (see page 578)	This is constant IBSQLMonitor.WM_MIN_IBSQL_MONITOR.

14.5.1 IBSQLMonitor.WM_IBSQL_SQL_EVENT Constant

Pascal

```
WM_IBSQL_SQL_EVENT = WM_MIN_IBSQL_MONITOR + 1;
```

C++

```
WM_IBSQL_SQL_EVENT = WM_MIN_IBSQL_MONITOR + 1;
```

File

IBSQLMonitor

Description

This is constant IBSQLMonitor.WM_IBSQL_SQL_EVENT.

14.5.2 IBSQLMonitor.WM_MAX_IBSQL_MONITOR Constant

Pascal

```
WM_MAX_IBSQL_MONITOR = WM_USER + 512;
```

C++

```
WM_MAX_IBSQL_MONITOR = WM_USER + 512;
```

File

IBSQLMonitor

Description

This is constant IBSQLMonitor.WM_MAX_IBSQL_MONITOR.

14.5.3 IBSQLMonitor.WM_MIN_IBSQL_MONITOR Constant

Pascal

```
WM_MIN_IBSQL_MONITOR = WM_USER;
```

C++

```
WM_MIN_IBSQL_MONITOR = WM_USER;
```

File

IBSQLMonitor

Description

This is constant IBSQLMonitor.WM_MIN_IBSQL_MONITOR.

15 IBStoredProc Namespace

This is namespace IBStoredProc.

Topics

Name	Description
Classes (see page 582)	The following table lists classes in this documentation.

Classes

	Name	Description
	TIBStoredProc (see page 582)	TIBStoredProc encapsulates a stored procedure on a database server.

15.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBStoredProc (see page 582)	TIBStoredProc encapsulates a stored procedure on a database server.

15.1.1 TIBStoredProc Class

TIBStoredProc encapsulates a stored procedure on a database server.

Class Hierarchy

Pascal

```
TIBStoredProc = class(TIBCustomDataSet);
```

C++

```
class TIBStoredProc : public TIBCustomDataSet;
```

File

IBStoredProc

Description

Use a TIBStoredProc object when a client application must use a stored procedure on a database server. A stored procedure is a grouped set of statements, stored as part of a database server's metadata (just like tables, indexes, and domains), that performs a frequently repeated, database-related task on the server and passes results to the client.

Many stored procedures require a series of input arguments, or parameters, that are used during processing. TIBStoredProc provides a Params (see page 593) property that enables an application to set these parameters before executing the stored procedure.

TIBStoredProc reuses the Params (see page 593) property to hold the results returned by a stored procedure. Params (see page 593) is an array of values. A stored procedure will return a single set of values, or none at all.

Note: TIBStoredProc components should be used with InterBase Execute stored procedures only. To use InterBase Select procedures, use TIBQuery or TIBDataSet (see page 114). Since Execute stored procedures do not return result sets, never use Open or Active on the TIBStoredProc component. Use ExecProc (see page 588) instead.

See Also

Types of Datasets

Getting Started with InterBase Express

15.1.1.1 TIBStoredProc Members

The following tables list the members exposed by TIBStoredProc.

Protected Methods

	Name	Description
	ActivateConnection (see page 69)	This is ActivateConnection, a member of class TIBCustomDataSet.
	ActivateTransaction (see page 69)	This is ActivateTransaction, a member of class TIBCustomDataSet.
V	AllocRecordBuffer (see page 69)	This is AllocRecordBuffer, a member of class TIBCustomDataSet.
	CheckDatasetClosed (see page 71)	This is CheckDatasetClosed, a member of class TIBCustomDataSet.
	CheckDatasetOpen (see page 71)	This is CheckDatasetOpen, a member of class TIBCustomDataSet.
	CheckNotUniDirectional (see page 71)	This is CheckNotUniDirectional, a member of class TIBCustomDataSet.
V	ClearCalcFields (see page 71)	This is ClearCalcFields, a member of class TIBCustomDataSet.
	ConstraintsStored (see page 72)	This is ConstraintsStored, a member of class TIBCustomDataSet.
V	CreateFields (see page 72)	This is CreateFields, a member of class TIBCustomDataSet.
	DeactivateTransaction (see page 73)	This is DeactivateTransaction, a member of class TIBCustomDataSet.
V	DefineProperties (see page 588)	This is DefineProperties, a member of class TIBStoredProc.
V	Disconnect (see page 588)	This is Disconnect, a member of class TIBStoredProc.
V	DoBeforeDelete (see page 73)	This is DoBeforeDelete, a member of class TIBCustomDataSet.
V	DoBeforeEdit (see page 73)	This is DoBeforeEdit, a member of class TIBCustomDataSet.
V	DoBeforeInsert (see page 74)	This is DoBeforeInsert, a member of class TIBCustomDataSet.
V	DoOnNewRecord (see page 74)	This is DoOnNewRecord, a member of class TIBCustomDataSet.
V	FreeRecordBuffer (see page 75)	This is FreeRecordBuffer, a member of class TIBCustomDataSet.
	GetActiveBuf (see page 75)	This is GetActiveBuf, a member of class TIBCustomDataSet.
V	GetBookmarkData (see page 75)	This is GetBookmarkData, a member of class TIBCustomDataSet.
V	GetBookmarkFlag (see page 75)	This is GetBookmarkFlag, a member of class TIBCustomDataSet.
V	GetCanModify (see page 75)	This is GetCanModify, a member of class TIBCustomDataSet.
V	GetDataSource (see page 76)	This is GetDataSource, a member of class TIBCustomDataSet.
V	GetFieldClass (see page 76)	This is GetFieldClass, a member of class TIBCustomDataSet.
	GetParamsCount (see page 589)	This is GetParamsCount, a member of class TIBStoredProc.
V	GetRecNo (see page 77)	This is GetRecNo, a member of class TIBCustomDataSet.
V	GetRecord (see page 77)	This is GetRecord, a member of class TIBCustomDataSet.
V	GetRecordCount (see page 78)	This is GetRecordCount, a member of class TIBCustomDataSet.
V	GetRecordSize (see page 78)	This is GetRecordSize, a member of class TIBCustomDataSet.
V	InitRecord (see page 78)	This is InitRecord, a member of class TIBCustomDataSet.
V	InternalAddRecord (see page 78)	This is InternalAddRecord, a member of class TIBCustomDataSet.
V	InternalBatchInput (see page 79)	This is InternalBatchInput, a member of class TIBCustomDataSet.
V	InternalBatchOutput (see page 79)	This is InternalBatchOutput, a member of class TIBCustomDataSet.
V	InternalCancel (see page 79)	This is InternalCancel, a member of class TIBCustomDataSet.
V	InternalClose (see page 79)	This is InternalClose, a member of class TIBCustomDataSet.
V	InternalDelete (see page 79)	This is InternalDelete, a member of class TIBCustomDataSet.
V	InternalExecQuery (see page 80)	This is InternalExecQuery, a member of class TIBCustomDataSet.
V	InternalFirst (see page 80)	This is InternalFirst, a member of class TIBCustomDataSet.
V	InternalGotoBookmark (see page 80)	This is InternalGotoBookmark, a member of class TIBCustomDataSet.
V	InternalHandleException (see page 80)	This is InternalHandleException, a member of class TIBCustomDataSet.
V	InternalInitFieldDefs (see page 81)	This is InternalInitFieldDefs, a member of class TIBCustomDataSet.
V	InternalInitRecord (see page 81)	This is InternalInitRecord, a member of class TIBCustomDataSet.
V	InternalInsert (see page 81)	This is InternalInsert, a member of class TIBCustomDataSet.
V	InternalLast (see page 81)	This is InternalLast, a member of class TIBCustomDataSet.
V	InternalOpen (see page 589)	This is InternalOpen, a member of class TIBStoredProc.
V	InternalPost (see page 82)	This is InternalPost, a member of class TIBCustomDataSet.
V	InternalPrepare (see page 82)	This is InternalPrepare, a member of class TIBCustomDataSet.
V	InternalRefresh (see page 82)	This is InternalRefresh, a member of class TIBCustomDataSet.
V	InternalRefreshRow (see page 82)	This is InternalRefreshRow, a member of class TIBCustomDataSet.
V	InternalSetParamsFromCursor (see page 83)	This is InternalSetParamsFromCursor, a member of class TIBCustomDataSet.
V	InternalSetToRecord (see page 83)	This is InternalSetToRecord, a member of class TIBCustomDataSet.
V	InternalUnPrepare (see page 83)	This is InternalUnPrepare, a member of class TIBCustomDataSet.
V	IsCursorOpen (see page 83)	This is IsCursorOpen, a member of class TIBCustomDataSet.

V	Loaded (see page 84)	This is Loaded, a member of class TIBCustomDataSet.
V	PSEndTransaction (see page 86)	This is PSEndTransaction, a member of class TIBCustomDataSet.
V	PSExecute (see page 590)	This is PSExecute, a member of class TIBStoredProc.
V	PSExecuteStatement (see page 86)	This is PSExecuteStatement, a member of class TIBCustomDataSet.
V	PSGetParams (see page 590)	This is PSGetParams, a member of class TIBStoredProc.
V	PSGetQuoteChar (see page 87)	This is PSGetQuoteChar, a member of class TIBCustomDataSet.
V	PSGetTableName (see page 590)	This is PSGetTableName, a member of class TIBStoredProc.
V	PSGetUpdateException (see page 87)	This is PSGetUpdateException, a member of class TIBCustomDataSet.
V	PSInTransaction (see page 87)	This is PSInTransaction, a member of class TIBCustomDataSet.
V	PSIsSQLBased (see page 88)	This is PSIsSQLBased, a member of class TIBCustomDataSet.
V	PSIsSQLSupported (see page 88)	This is PSIsSQLSupported, a member of class TIBCustomDataSet.
V	PSReset (see page 88)	This is PSReset, a member of class TIBCustomDataSet.
V	PSSetCommandText (see page 591)	This is PSSetCommandText, a member of class TIBStoredProc.
V	PSSetParams (see page 591)	This is PSSetParams, a member of class TIBStoredProc.
V	PSStartTransaction (see page 88)	This is PSStartTransaction, a member of class TIBCustomDataSet.
V	PSUpdateRecord (see page 88)	This is PSUpdateRecord, a member of class TIBCustomDataSet.
	ReQuery (see page 89)	This is ReQuery, a member of class TIBCustomDataSet.
V	SetActive (see page 89)	This is SetActive, a member of class TIBCustomDataSet.
V	SetBookmarkData (see page 90)	This is SetBookmarkData, a member of class TIBCustomDataSet.
V	SetBookmarkFlag (see page 90)	This is SetBookmarkFlag, a member of class TIBCustomDataSet.
	SetCachedUpdates (see page 90)	This is SetCachedUpdates, a member of class TIBCustomDataSet.
	SetDataSource (see page 90)	This is SetDataSource, a member of class TIBCustomDataSet.
V	SetFieldData (see page 90)	This is the overview for the SetFieldData method overload.
V	SetFiltered (see page 591)	This is SetFiltered, a member of class TIBStoredProc.
	SetPrepare (see page 591)	This is SetPrepare, a member of class TIBStoredProc.
	SetPrepared (see page 591)	This is SetPrepared, a member of class TIBStoredProc.
	SetProcName (see page 592)	This is SetProcName, a member of class TIBStoredProc.
V	SetRecNo (see page 91)	This is SetRecNo, a member of class TIBCustomDataSet.

Protected Properties

	Name	Description
	BufferChunks (see page 98)	Specifies the dataset buffer chunk size.
	CachedUpdates (see page 98)	Specifies whether cached updates are enabled for a dataset.
	DeleteSQL (see page 99)	Holds the SQL statement used to delete rows from the dataset.
	GeneratorField (see page 100)	Specifies whether the dataset should refresh its data every time it posts a record.
	InsertSQL (see page 100)	Holds the SQL statement used to insert rows into the dataset.
R	InternalPrepared (see page 100)	Indicates whether the dataset has prepared its queries for execution.
R	LiveMode (see page 101)	Indicates what types of edits, if any, can be made to the data in the dataset.
	ModifySQL (see page 101)	Provides the ability to access the SQL object encapsulating the ModifySQL statement.
	ParamCheck (see page 102)	Specifies whether the parameter list for a query is regenerated if the SQL property changes at runtime.
R	QDelete (see page 103)	Provides the ability to directly access the SQL object encapsulating the DeleteSQL (see page 99) statement.
R	QInsert (see page 104)	Provides the ability to directly access the SQL object encapsulating the InsertSQL (see page 100) statement.
R	QModify (see page 104)	Provides the ability to directly access the SQL object encapsulating the ModifySQL (see page 101) statement.
R	QRefresh (see page 104)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL (see page 105) statement.
R	QSelect (see page 104)	Provides the ability to directly access the SQL object encapsulating the SelectSQL (see page 105) statement.
	RefreshSQL (see page 105)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.
	SelectSQL (see page 105)	Provides the ability to directly access the SQL object encapsulating the SelectSQL statement.
R	SelectStmtHandle (see page 105)	Gets the handle for the SELECT statement.

	SQLParams (see page 106)	Specifies the parameters to use for any parameterized queries.
	StatementType (see page 106)	Returns the statement type of the QSelect (see page 104) query.
	UniDirectional (see page 107)	Determines whether bidirectional cursors are enabled for the dataset.
	UpdateMode (see page 107)	Indicates how records are located when updating records.

Public Methods

	Name	Description
	ApplyUpdates (see page 69)	Writes a dataset's pending cached updates to the database.
	BookmarkValid (see page 70)	Indicates the validity of a specified bookmark.
	CachedUpdateStatus (see page 70)	Returns the status of the cached updates.
	CancelUpdates (see page 70)	Clears all pending cached updates from the cache.
	CompareBookmarks (see page 71)	Compares two bookmarks.
	CopyParams (see page 588)	Copies a stored procedure's parameters into another parameter list.
	Create (see page 586)	Creates an instance of a stored procedure component.
	CreateBlobStream (see page 72)	Returns a TIBDSBlobStream (see page 138) object for reading or writing the data in a specified blob field.
	Current (see page 73)	Returns the descriptor area that holds information about the current query.
	Destroy (see page 587)	Destroys the instance of a stored procedure.
	ExecProc (see page 588)	Executes the stored procedure on the server.
	FetchAll (see page 74)	Retrieves all records from the current cursor position to the end of the file and stores them locally.
	GetCurrentRecord (see page 76)	Retrieves the current record into a buffer.
	GetFieldData (see page 76)	This is the overview for the GetFieldData method overload.
	IsSequenced (see page 83)	IsSequenced indicates whether sequence numbers are available for database records.
	Locate (see page 84)	Searches the dataset for a specified record and makes that record the current record.
	LocateNext (see page 85)	Searches the dataset for the record after a specified record and makes that record the current record.
	Lookup (see page 85)	Retrieves field values from a record that matches specified search values.
	ParamByName (see page 589)	Accesses parameter information based on a specified parameter name.
	Post (see page 86)	Writes a modified record to the database server.
	Prepare (see page 590)	Prepares a stored procedure for execution.
	RecordModified (see page 89)	Sets the record to modified or unmodified.
	RevertRecord (see page 89)	Restores the current record in the dataset to an unmodified state when cached updates are enabled.
	SQLType (see page 91)	Returns the type of query to be executed.
	Undelete (see page 92)	Restores a record deleted from the dataset.
	UnPrepare (see page 592)	Frees the resources allocated for a previously prepared stored procedure.
	UpdateStatus (see page 92)	Reports the update status for the current record.

Public Properties

	Name	Description
	DBHandle (see page 99)	Specifies the database handle for the dataset.
	ParamCount (see page 593)	Indicates the number of parameters for the stored procedure component.
	Plan (see page 103)	Indicates the PLAN portion of the dataset's SELECT query.
	Prepared (see page 594)	Determines whether or not a stored procedure is prepared for execution.
	RowsAffected (see page 105)	Indicates the number of rows operated on by the latest query execution.
	StmtHandle (see page 594)	Identifies the statement handle for the stored procedure.
	StoredProcedureNames (see page 594)	Returns a list of stored procedures in the database.
	TRHandle (see page 106)	Specifies the transaction handle for the dataset.
	UpdateObject (see page 107)	Specifies the update object component used to update a read-only result set.
	UpdateRecordTypes (see page 108)	Specifies the type of records visible in a dataset when cached updates are enabled.
	UpdatesPending (see page 108)	Indicates whether the cached updates buffer contains records that are not yet applied.

Published Events

	Name	Description
	AfterDatabaseDisconnect (see page 595)	Occurs immediately after the connection to the associated database is closed.
	AfterTransactionEnd (see page 595)	Occurs immediately after a transaction that includes this dataset's query ends.
	BeforeDatabaseDisconnect (see page 596)	Occurs immediately before the connection to the associated database is closed.
	BeforeTransactionEnd (see page 596)	Occurs immediately before a transaction that includes this dataset's query ends.
	DatabaseFree (see page 596)	Occurs when the database associated with this dataset is freed.
	OnUpdateError (see page 110)	Occurs if an exception is generated when cached updates are applied to a database.
	OnUpdateRecord (see page 111)	Occurs when cached updates are applied to a record.
	TransactionFree (see page 596)	Occurs when the transaction that includes this dataset's query is freed.

Published Properties

	Name	Description
	AfterCancel (see page 94)	This is AfterCancel, a member of class TIBCustomDataSet.
	AfterClose (see page 94)	This is AfterClose, a member of class TIBCustomDataSet.
	AfterDelete (see page 94)	This is AfterDelete, a member of class TIBCustomDataSet.
	AfterEdit (see page 94)	This is AfterEdit, a member of class TIBCustomDataSet.
	AfterInsert (see page 95)	This is AfterInsert, a member of class TIBCustomDataSet.
	AfterOpen (see page 95)	This is AfterOpen, a member of class TIBCustomDataSet.
	AfterPost (see page 95)	This is AfterPost, a member of class TIBCustomDataSet.
	AfterRefresh (see page 95)	This is AfterRefresh, a member of class TIBCustomDataSet.
	AfterScroll (see page 95)	This is AfterScroll, a member of class TIBCustomDataSet.
	AutoCalcFields (see page 96)	This is AutoCalcFields, a member of class TIBCustomDataSet.
	BeforeCancel (see page 96)	This is BeforeCancel, a member of class TIBCustomDataSet.
	BeforeClose (see page 96)	This is BeforeClose, a member of class TIBCustomDataSet.
	BeforeDelete (see page 96)	This is BeforeDelete, a member of class TIBCustomDataSet.
	BeforeEdit (see page 97)	This is BeforeEdit, a member of class TIBCustomDataSet.
	BeforeInsert (see page 97)	This is BeforeInsert, a member of class TIBCustomDataSet.
	BeforeOpen (see page 97)	This is BeforeOpen, a member of class TIBCustomDataSet.
	BeforePost (see page 97)	This is BeforePost, a member of class TIBCustomDataSet.
	BeforeRefresh (see page 97)	This is BeforeRefresh, a member of class TIBCustomDataSet.
	BeforeScroll (see page 98)	This is BeforeScroll, a member of class TIBCustomDataSet.
	Database (see page 99)	Identifies the database component for which this dataset represents one or more tables.
	Filtered (see page 592)	Specifies whether filtering is active for a dataset.
	ForcedRefresh (see page 99)	Specifies whether the dataset should refresh its data every time it posts a record.
	ObjectView (see page 101)	This is ObjectView, a member of class TIBCustomDataSet.
	OnCalcFields (see page 101)	This is OnCalcFields, a member of class TIBCustomDataSet.
	OnDeleteError (see page 102)	This is OnDeleteError, a member of class TIBCustomDataSet.
	OnEditError (see page 102)	This is OnEditError, a member of class TIBCustomDataSet.
	OnFilterRecord (see page 593)	This is OnFilterRecord, a member of class TIBStoredProc.
	OnNewRecord (see page 102)	This is OnNewRecord, a member of class TIBCustomDataSet.
	OnPostError (see page 102)	This is OnPostError, a member of class TIBCustomDataSet.
	Params (see page 593)	Stores the input and output parameters for a stored procedure.
	StoredProcName (see page 595)	Identifies the name of the stored procedure on the server for which this object is an encapsulation.
	Transaction (see page 106)	Identifies the transaction under which the query executes.

15.1.1.2 TIBStoredProc.Create Constructor

Creates an instance of a stored procedure component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBStoredProc(TComponent * AOwner);
```

Description

Call Create to instantiate a stored procedure declared in an application. Create calls its inherited Create constructor, creates an empty parameter list for the newly instantiated stored procedure, and initializes its parameter, server, and record buffers to nil.

15.1.1.3 TIBStoredProc.Destroy Destructor

Destroys the instance of a stored procedure.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBStoredProc();
```

Description

Do not call Destroy directly. Instead call Free to verify that the stored procedure is not already freed before calling Destroy. Destroy disconnects from the server, frees the parameter list, and calls its inherited Destroy destructor.

15.1.1.4 TIBStoredProc Methods

The methods of the TIBStoredProc class are listed here.

Protected Methods

	Name	Description
	DefineProperties (see page 588)	This is DefineProperties, a member of class TIBStoredProc.
	Disconnect (see page 588)	This is Disconnect, a member of class TIBStoredProc.
	GetParamsCount (see page 589)	This is GetParamsCount, a member of class TIBStoredProc.
	InternalOpen (see page 589)	This is InternalOpen, a member of class TIBStoredProc.
	PSExecute (see page 590)	This is PSExecute, a member of class TIBStoredProc.
	PSGetParams (see page 590)	This is PSGetParams, a member of class TIBStoredProc.
	PSGetTableName (see page 590)	This is PSGetTableName, a member of class TIBStoredProc.
	PSSetCommandText (see page 591)	This is PSSetCommandText, a member of class TIBStoredProc.
	PSSetParams (see page 591)	This is PSSetParams, a member of class TIBStoredProc.
	SetFiltered (see page 591)	This is SetFiltered, a member of class TIBStoredProc.
	SetPrepare (see page 591)	This is SetPrepare, a member of class TIBStoredProc.
	SetPrepared (see page 591)	This is SetPrepared, a member of class TIBStoredProc.
	SetProcName (see page 592)	This is SetProcName, a member of class TIBStoredProc.

Public Methods

	Name	Description
	CopyParams (see page 588)	Copies a stored procedure's parameters into another parameter list.
	ExecProc (see page 588)	Executes the stored procedure on the server.
	ParamByName (see page 589)	Accesses parameter information based on a specified parameter name.
	Prepare (see page 590)	Prepares a stored procedure for execution.
	UnPrepare (see page 592)	Frees the resources allocated for a previously prepared stored procedure.

15.1.1.4.1 TIBStoredProc.CopyParams Method

Copies a stored procedure's parameters into another parameter list.

Pascal

```
procedure CopyParams(Value: TParams);
```

C++

```
__fastcall CopyParams(TParams Value);
```

Description

Call CopyParams to copy this stored procedure's parameters into a separate parameter list object. Value is the parameter list into which to assign this stored procedure's parameters. Value can be the parameter list of another stored procedure. For example:

```
TIBStoredProc1.CopyParams(TIBStoredProc2.Params);
    TIBStoredProc1->CopyParams(TIBStoredProc2->Params);
```

If the stored procedure is not prepared when an application calls CopyParams, CopyParams calls Prepare (see page 590) before assigning the parameters to the target parameters list, and then calls UnPrepare (see page 592) to return the stored procedure to its previous state.

15.1.1.4.2 TIBStoredProc.DefineProperties Method

Pascal

```
procedure DefineProperties(Filer: TFiler); override;
```

C++

```
virtual __fastcall DefineProperties(TFiler Filer);
```

Description

This is DefineProperties, a member of class TIBStoredProc.

15.1.1.4.3 TIBStoredProc.Disconnect Method

Pascal

```
procedure Disconnect; override;
```

C++

```
virtual __fastcall Disconnect();
```

Description

This is Disconnect, a member of class TIBStoredProc.

15.1.1.4.4 TIBStoredProc.ExecProc Method

Executes the stored procedure on the server.

Pascal

```
procedure ExecProc;
```

C++

```
__fastcall ExecProc();
```

Description

Call ExecProc to execute a stored procedure on the server. Before calling ExecProc:

1. **1** Provide any input parameters in the Params (see page 593) property. At design time, a developer can provide parameters using the Parameters editor. At runtime an application must access Params (see page 593) directly.
2. **2** Call Prepare (see page 590) to bind the parameters.

If a stored procedure returns output parameters, they are stored in the Params (see page 593) property when ExecProc returns control to the application. An application can access the output parameters by indexing into the Params (see page 593) list, or by using the ParamByName (see page 589) method.

15.1.1.4.5 TIBStoredProc.GetParamsCount Method

Pascal

```
function GetParamsCount: Word;
```

C++

```
__fastcall Word GetParamsCount();
```

Description

This is GetParamsCount, a member of class TIBStoredProc.

15.1.1.4.6 TIBStoredProc.InternalOpen Method

Pascal

```
procedure InternalOpen; override;
```

C++

```
virtual __fastcall InternalOpen();
```

Description

This is InternalOpen, a member of class TIBStoredProc.

15.1.1.4.7 TIBStoredProc.ParamByName Method

Accesses parameter information based on a specified parameter name.

Pascal

```
function ParamByName(const Value: string): TParam;
```

C++

```
__fastcall TParam ParamByName(const UnicodeString Value);
```

Description

Call ParamByName to return parameter information for a specific parameter based on its name. Value is the name of the parameter for which to retrieve information. Typically ParamByName is used to set an input parameter's value at runtime, or to retrieve the value of an output parameter. The following command line assigns the value "Jane Smith" as the value for the parameter named Contact:

```
StoredProc1.ParamByName('Contact').AsString := 'Jane Smith';
    StoredProc1->ParamByName("Contact")->AsString = "Jane Smith";
```

15.1.1.4.8 TIBStoredProc.Prepare Method

Prepares a stored procedure for execution.

Pascal

```
procedure Prepare;
```

C++

```
__fastcall Prepare();
```

Description

Call Prepare to bind a stored procedure's parameters before calling ExecProc (see page 588) to execute the procedure. Prepare readies a stored procedure's parameters and informs the server of the stored procedure's readiness. These steps allocate system resources and optimize the query for server performance.

Note: A stored procedure is automatically prepared if it is unprepared when the application calls ExecProc (see page 588). After execution, the stored procedure is then automatically unprepared. If a stored procedure will be executed more than once, it is more efficient for an application to call Prepare explicitly once to avoid repeated and unnecessary preparing and unpreparing of the stored procedure, and then call UnPrepare (see page 592) when the stored procedure is no longer needed.

15.1.1.4.9 TIBStoredProc.PSExecute Method

Pascal

```
procedure PSExecute; override;
```

C++

```
virtual __fastcall PSExecute();
```

Description

This is PSExecute, a member of class TIBStoredProc.

15.1.1.4.10 TIBStoredProc.PSGetParams Method

Pascal

```
function PSGetParams: TParams; override;
```

C++

```
virtual __fastcall TParams PSGetParams();
```

Description

This is PSGetParams, a member of class TIBStoredProc.

15.1.1.4.11 TIBStoredProc.PSGetTableName Method

Pascal

```
function PSGetTableName: string; override;
```

C++

```
virtual __fastcall UnicodeString PSGetTableName();
```

Description

This is PSGetTableName, a member of class TIBStoredProc.

15.1.1.4.12 TIBStoredProc.PSSetCommandText Method

Pascal

```
procedure PSSetCommandText(const CommandText: string); override;
```

C++

```
virtual __fastcall PSSetCommandText(const UnicodeString CommandText);
```

Description

This is PSSetCommandText, a member of class TIBStoredProc.

15.1.1.4.13 TIBStoredProc.PSSetParams Method

Pascal

```
procedure PSSetParams(AParams: TParams); override;
```

C++

```
virtual __fastcall PSSetParams(TParams AParams);
```

Description

This is PSSetParams, a member of class TIBStoredProc.

15.1.1.4.14 TIBStoredProc.SetFiltered Method

Pascal

```
procedure SetFiltered(Value: Boolean); override;
```

C++

```
virtual __fastcall SetFiltered(Boolean Value);
```

Description

This is SetFiltered, a member of class TIBStoredProc.

15.1.1.4.15 TIBStoredProc.SetPrepare Method

Pascal

```
procedure SetPrepare(Value: Boolean);
```

C++

```
__fastcall SetPrepare(Boolean Value);
```

Description

This is SetPrepare, a member of class TIBStoredProc.

15.1.1.4.16 TIBStoredProc.SetPrepared Method

Pascal

```
procedure SetPrepared(Value: Boolean);
```

C++

```
__fastcall SetPrepared(Boolean Value);
```

Description

This is SetPrepared, a member of class TIBStoredProc.

15.1.1.4.17 TIBStoredProc.SetProcName Method

Pascal

```
procedure SetProcName(Value: string);
```

C++

```
__fastcall SetProcName(UnicodeString Value);
```

Description

This is SetProcName, a member of class TIBStoredProc.

15.1.1.4.18 TIBStoredProc.UnPrepare Method

Frees the resources allocated for a previously prepared stored procedure.

Pascal

```
procedure UnPrepare;
```

C++

```
__fastcall UnPrepare();
```

Description

Call UnPrepare to free the resources allocated for a previously prepared stored procedure on the server and client sides.

15.1.1.5 TIBStoredProc Properties

The properties of the TIBStoredProc class are listed here.

Public Properties

	Name	Description
R	ParamCount (see page 593)	Indicates the number of parameters for the stored procedure component.
	Prepared (see page 594)	Determines whether or not a stored procedure is prepared for execution.
R	StmtHandle (see page 594)	Identifies the statement handle for the stored procedure.
R	StoredProcedureNames (see page 594)	Returns a list of stored procedures in the database.

Published Properties

	Name	Description
	Filtered (see page 592)	Specifies whether filtering is active for a dataset.
	OnFilterRecord (see page 593)	This is OnFilterRecord, a member of class TIBStoredProc.
	Params (see page 593)	Stores the input and output parameters for a stored procedure.
	StoredProcName (see page 595)	Identifies the name of the stored procedure on the server for which this object is an encapsulation.

15.1.1.5.1 TIBStoredProc.Filtered Property

Specifies whether filtering is active for a dataset.

Pascal

```
property Filtered;
```

C++

```
__property Filtered;
```

Description

Check Filtered to determine whether dataset filtering is in effect. If Filtered is true, then filtering is active. Otherwise Filtered is false. To apply filter conditions specified in the Filter property or the OnFilterRecord (see page 593) event handler, set Filtered to true. Filtered is set to false by default.

Note: When filtering is enabled, user edits to a record may mean that the record no longer meets a filter's test condition. The next time the record is retrieved from the dataset while the filter is in effect, the record may seem to disappear. If that happens, the next record that passes the filter condition becomes the current record.

15.1.1.5.2 TIBStoredProc.OnFilterRecord Property

Pascal

```
property OnFilterRecord;
```

C++

```
__property OnFilterRecord;
```

Description

This is OnFilterRecord, a member of class TIBStoredProc.

15.1.1.5.3 TIBStoredProc.ParamCount Property

Indicates the number of parameters for the stored procedure component.

Pascal

```
property ParamCount: Word;
```

C++

```
__property Word ParamCount;
```

Description

Examine ParamCount to determine the number of parameters currently stored in the Params (see page 593) property.

15.1.1.5.4 TIBStoredProc.Params Property

Stores the input and output parameters for a stored procedure.

Pascal

```
property Params: TParams;
```

C++

```
__property TParams Params;
```

Description

Access Params at runtime to set input parameter names, values, and data types dynamically (at design time use the Parameters editor to set parameter information). Params is an array of parameter values.

An application can also access Params after executing a stored procedure to retrieve the output parameters returned to the

procedure by the server.

15.1.1.5.5 TIBStoredProc.Prepared Property

Determines whether or not a stored procedure is prepared for execution.

Pascal

```
property Prepared: Boolean;
```

C++

```
__property Boolean Prepared;
```

Description

Examine Prepared to determine if a stored procedure is already prepared for execution. If Prepared is true, the stored procedure is prepared, and if Prepared is false, the procedure is not prepared. A stored procedure must be prepared before it can be executed.

Note: A stored procedure is automatically prepared if it is unprepared when the application calls ExecProc (see page 588). After execution, the stored procedure is then automatically unprepared. If a procedure will be executed a number of times, it is more efficient for the application to prepare the stored procedure once, and unprepare it when it is no longer needed.

An application can change the current setting of Prepared to prepare or unprepare a stored procedure. If Prepared is true, setting it to false calls the UnPrepare (see page 592) method to unprepare the stored procedure. If Prepared is false, setting it to true calls the Prepare (see page 590) method to prepare the procedure. Generally, however, it is better programming practice to call Prepare (see page 590) and UnPrepare (see page 592) directly. These methods automatically update the Prepared property.

15.1.1.5.6 TIBStoredProc.StmtHandle Property

Identifies the statement handle for the stored procedure.

Pascal

```
property StmtHandle: TISC_STMT_HANDLE;
```

C++

```
__property TISC_STMT_HANDLE StmtHandle;
```

Description

Retrieve StmtHandle if an application makes a direct call to the InterBase server, bypassing the methods of TIBStoredProc (see page 582). Some API calls require a statement handle as a parameter. Under all other circumstances an application does not need to access this property.

15.1.1.5.7 TIBStoredProc.StoredProcedureNames Property

Returns a list of stored procedures in the database.

Pascal

```
property StoredProcedureNames: TStrings;
```

C++

```
__property TStrings StoredProcedureNames;
```

Description

StoredProcedureNames is an internal property used to list the stored procedures available in the database.

15.1.1.5.8 TIBStoredProc.StoredProcName Property

Identifies the name of the stored procedure on the server for which this object is an encapsulation.

Pascal

```
property StoredProcName: string;
```

C++

```
__property UnicodeString StoredProcName;
```

Description

Set StoredProcName to specify the name of the stored procedure to call on the server. If StoredProcName does not match the name of an existing stored procedure on the server, then when the application attempts to prepare the procedure prior to execution, an exception is raised.

15.1.1.6 TIBStoredProc Events

The events of the TIBStoredProc class are listed here.

Published Events

	Name	Description
	AfterDatabaseDisconnect (see page 595)	Occurs immediately after the connection to the associated database is closed.
	AfterTransactionEnd (see page 595)	Occurs immediately after a transaction that includes this dataset's query ends.
	BeforeDatabaseDisconnect (see page 596)	Occurs immediately before the connection to the associated database is closed.
	BeforeTransactionEnd (see page 596)	Occurs immediately before a transaction that includes this dataset's query ends.
	DatabaseFree (see page 596)	Occurs when the database associated with this dataset is freed.
	TransactionFree (see page 596)	Occurs when the transaction that includes this dataset's query is freed.

15.1.1.6.1 TIBStoredProc.AfterDatabaseDisconnect Event

Occurs immediately after the connection to the associated database is closed.

Pascal

```
property AfterDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterDatabaseDisconnect;
```

Description

Write an AfterDatabaseDisconnect event handler to respond when the dataset closes its connection to the database specified by the Database property.

15.1.1.6.2 TIBStoredProc.AfterTransactionEnd Event

Occurs immediately after a transaction that includes this dataset's query ends.

Pascal

```
property AfterTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterTransactionEnd;
```

Description

Write an AfterTransactionEnd event handler to respond when the transaction specified by the Transaction property ends.

15.1.1.6.3 TIBStoredProc.BeforeDatabaseDisconnect Event

Occurs immediately before the connection to the associated database is closed.

Pascal

```
property BeforeDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeDatabaseDisconnect;
```

Description

Write a BeforeDatabaseDisconnect event handler to respond when the dataset is about to close its connection to the database specified by the Database property.

15.1.1.6.4 TIBStoredProc.BeforeTransactionEnd Event

Occurs immediately before a transaction that includes this dataset's query ends.

Pascal

```
property BeforeTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeTransactionEnd;
```

Description

Write a BeforeTransactionEnd event handler to respond when the transaction specified by the Transaction property is about to end.

15.1.1.6.5 TIBStoredProc.DatabaseFree Event

Occurs when the database associated with this dataset is freed.

Pascal

```
property DatabaseFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent DatabaseFree;
```

Description

Write a DatabaseFree event handler to respond when the database specified by the Database property is freed.

15.1.1.6.6 TIBStoredProc.TransactionFree Event

Occurs when the transaction that includes this dataset's query is freed.

Pascal

```
property TransactionFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent TransactionFree;
```

Description

Write a TransactionFree event handler to respond when the transaction specified by the Transaction property is freed.

16 IBTable Namespace

This is namespace IBTable.

Topics

Name	Description
Classes (see page 600)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 626)	The following table lists structs, records, enums in this documentation.
Types (see page 627)	The following table lists types in this documentation.

Classes

	Name	Description
	TIBTable (see page 600)	TIBTable is a dataset component that encapsulates a database table.

Structs, Records, Enums

	Name	Description
	TIBTableType (see page 626)	This is record IBTable.TIBTableType.

Types

Name	Description
TIBTableTypes (see page 627)	TIBTableTypes and TIBTableTypes indicate the types of InterBase tables and views other than user tables.
TIndexName (see page 627)	This is type IBTable.TIndexName.

16.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBTable (see page 600)	TIBTable is a dataset component that encapsulates a database table.

16.1.1 TIBTable Class

TIBTable is a dataset component that encapsulates a database table.

Class Hierarchy

Pascal

```
TIBTable = class(TIBCustomDataSet);
```

C++

```
class TIBTable : public TIBCustomDataSet;
```

File

IBTable

Description

Use TIBTable to access data in a single table or view. TIBTable provides direct access to every record and field in an underlying InterBase database table. A table component can also work with a subset of records within a database table using filters.

See Also

Types of Datasets

Getting Started with InterBase Express

16.1.1.1 TIBTable Members

The following tables list the members exposed by TIBTable.

Protected Methods

	Name	Description
	ActivateConnection (see page 69)	This is ActivateConnection, a member of class TIBCustomDataSet.
	ActivateTransaction (see page 69)	This is ActivateTransaction, a member of class TIBCustomDataSet.
	AllocRecordBuffer (see page 69)	This is AllocRecordBuffer, a member of class TIBCustomDataSet.
	CheckDatasetClosed (see page 71)	This is CheckDatasetClosed, a member of class TIBCustomDataSet.
	CheckDatasetOpen (see page 71)	This is CheckDatasetOpen, a member of class TIBCustomDataSet.
	CheckNotUniDirectional (see page 71)	This is CheckNotUniDirectional, a member of class TIBCustomDataSet.
	ClearCalcFields (see page 71)	This is ClearCalcFields, a member of class TIBCustomDataSet.
	ConstraintsStored (see page 72)	This is ConstraintsStored, a member of class TIBCustomDataSet.

	CreateFields (see page 72)	This is CreateFields, a member of class TIBCustomDataSet.
	DataEvent (see page 607)	This is DataEvent, a member of class TIBTable.
	DeactivateTransaction (see page 73)	This is DeactivateTransaction, a member of class TIBCustomDataSet.
	DefChanged (see page 607)	This is DefChanged, a member of class TIBTable.
	Disconnect (see page 73)	This is Disconnect, a member of class TIBCustomDataSet.
	DoBeforeDelete (see page 73)	This is DoBeforeDelete, a member of class TIBCustomDataSet.
	DoBeforeEdit (see page 73)	This is DoBeforeEdit, a member of class TIBCustomDataSet.
	DoBeforeInsert (see page 74)	This is DoBeforeInsert, a member of class TIBCustomDataSet.
	DoOnNewRecord (see page 608)	This is DoOnNewRecord, a member of class TIBTable.
	FreeRecordBuffer (see page 75)	This is FreeRecordBuffer, a member of class TIBCustomDataSet.
	GetActiveBuf (see page 75)	This is GetActiveBuf, a member of class TIBCustomDataSet.
	GetBookmarkData (see page 75)	This is GetBookmarkData, a member of class TIBCustomDataSet.
	GetBookmarkFlag (see page 75)	This is GetBookmarkFlag, a member of class TIBCustomDataSet.
	GetCanModify (see page 608)	This is GetCanModify, a member of class TIBTable.
	GetDataSource (see page 609)	This is GetDataSource, a member of class TIBTable.
	GetFieldClass (see page 76)	This is GetFieldClass, a member of class TIBCustomDataSet.
	GetIndexParams (see page 609)	This is GetIndexParams, a member of class TIBTable.
	GetRecNo (see page 77)	This is GetRecNo, a member of class TIBCustomDataSet.
	GetRecord (see page 77)	This is GetRecord, a member of class TIBCustomDataSet.
	GetRecordCount (see page 78)	This is GetRecordCount, a member of class TIBCustomDataSet.
	GetRecordSize (see page 78)	This is GetRecordSize, a member of class TIBCustomDataSet.
	InitFieldDefs (see page 610)	This is InitFieldDefs, a member of class TIBTable.
	InitRecord (see page 78)	This is InitRecord, a member of class TIBCustomDataSet.
	InternalAddRecord (see page 78)	This is InternalAddRecord, a member of class TIBCustomDataSet.
	InternalBatchInput (see page 79)	This is InternalBatchInput, a member of class TIBCustomDataSet.
	InternalBatchOutput (see page 79)	This is InternalBatchOutput, a member of class TIBCustomDataSet.
	InternalCancel (see page 79)	This is InternalCancel, a member of class TIBCustomDataSet.
	InternalClose (see page 610)	This is InternalClose, a member of class TIBTable.
	InternalDelete (see page 79)	This is InternalDelete, a member of class TIBCustomDataSet.
	InternalExecQuery (see page 80)	This is InternalExecQuery, a member of class TIBCustomDataSet.
	InternalFirst (see page 80)	This is InternalFirst, a member of class TIBCustomDataSet.
	InternalGotoBookmark (see page 80)	This is InternalGotoBookmark, a member of class TIBCustomDataSet.
	InternalHandleException (see page 80)	This is InternalHandleException, a member of class TIBCustomDataSet.
	InternalInitFieldDefs (see page 81)	This is InternalInitFieldDefs, a member of class TIBCustomDataSet.
	InternalInitRecord (see page 81)	This is InternalInitRecord, a member of class TIBCustomDataSet.
	InternalInsert (see page 81)	This is InternalInsert, a member of class TIBCustomDataSet.
	InternalLast (see page 81)	This is InternalLast, a member of class TIBCustomDataSet.
	InternalOpen (see page 610)	This is InternalOpen, a member of class TIBTable.
	InternalPost (see page 82)	This is InternalPost, a member of class TIBCustomDataSet.
	InternalPrepare (see page 82)	This is InternalPrepare, a member of class TIBCustomDataSet.
	InternalRefresh (see page 611)	This is InternalRefresh, a member of class TIBTable.
	InternalRefreshRow (see page 611)	This is InternalRefreshRow, a member of class TIBTable.
	InternalSetParamsFromCursor (see page 83)	This is InternalSetParamsFromCursor, a member of class TIBCustomDataSet.
	InternalSetToRecord (see page 83)	This is InternalSetToRecord, a member of class TIBCustomDataSet.
	InternalUnPrepare (see page 83)	This is InternalUnPrepare, a member of class TIBCustomDataSet.
	IsCursorOpen (see page 83)	This is IsCursorOpen, a member of class TIBCustomDataSet.
	Loaded (see page 84)	This is Loaded, a member of class TIBCustomDataSet.
	PSEndTransaction (see page 86)	This is PSEndTransaction, a member of class TIBCustomDataSet.
	PSExecuteStatement (see page 86)	This is PSExecuteStatement, a member of class TIBCustomDataSet.
	PSGetDefaultOrder (see page 612)	This is PSGetDefaultOrder, a member of class TIBTable.
	PSGetIndexDefs (see page 612)	This is PSGetIndexDefs, a member of class TIBTable.
	PSGetKeyFields (see page 612)	This is PSGetKeyFields, a member of class TIBTable.
	PSGetQuoteChar (see page 87)	This is PSGetQuoteChar, a member of class TIBCustomDataSet.
	PSGetTableName (see page 612)	This is PSGetTableName, a member of class TIBTable.

	PSGetUpdateException (see page 87)	This is PSGetUpdateException, a member of class TIBCustomDataSet.
	PSInTransaction (see page 87)	This is PSInTransaction, a member of class TIBCustomDataSet.
	PSIsSQLBased (see page 88)	This is PSIsSQLBased, a member of class TIBCustomDataSet.
	PSIsSQLSupported (see page 88)	This is PSIsSQLSupported, a member of class TIBCustomDataSet.
	PSReset (see page 88)	This is PSReset, a member of class TIBCustomDataSet.
	PSSetCommandText (see page 613)	This is PSSetCommandText, a member of class TIBTable.
	PSSetParams (see page 613)	This is PSSetParams, a member of class TIBTable.
	PSStartTransaction (see page 88)	This is PSStartTransaction, a member of class TIBCustomDataSet.
	PSUpdateRecord (see page 88)	This is PSUpdateRecord, a member of class TIBCustomDataSet.
	ReQuery (see page 89)	This is ReQuery, a member of class TIBCustomDataSet.
	SetActive (see page 89)	This is SetActive, a member of class TIBCustomDataSet.
	SetBookmarkData (see page 90)	This is SetBookmarkData, a member of class TIBCustomDataSet.
	SetBookmarkFlag (see page 90)	This is SetBookmarkFlag, a member of class TIBCustomDataSet.
	SetCachedUpdates (see page 90)	This is SetCachedUpdates, a member of class TIBCustomDataSet.
	SetDataSource (see page 90)	This is SetDataSource, a member of class TIBCustomDataSet.
	SetFieldData (see page 90)	This is the overview for the SetFieldData method overload.
	SetFiltered (see page 613)	This is SetFiltered, a member of class TIBTable.
	SetFilterOptions (see page 613)	This is SetFilterOptions, a member of class TIBTable.
	SetFilterText (see page 614)	This is SetFilterText, a member of class TIBTable.
	SetRecNo (see page 91)	This is SetRecNo, a member of class TIBCustomDataSet.
	UpdateIndexDefs (see page 614)	This is UpdateIndexDefs, a member of class TIBTable.

Protected Properties

	Name	Description
	DeleteSQL (see page 99)	Holds the SQL statement used to delete rows from the dataset.
	GeneratorField (see page 100)	Specifies whether the dataset should refresh its data every time it posts a record.
	InsertSQL (see page 100)	Holds the SQL statement used to insert rows into the dataset.
	InternalPrepared (see page 100)	Indicates whether the dataset has prepared its queries for execution.
	ModifySQL (see page 101)	Provides the ability to access the SQL object encapsulating the ModifySQL statement.
	ParamCheck (see page 102)	Specifies whether the parameter list for a query is regenerated if the SQL property changes at runtime.
	Params (see page 103)	Specifies the parameters to use for any parameterized queries.
	QDelete (see page 103)	Provides the ability to directly access the SQL object encapsulating the DeleteSQL (see page 99) statement.
	QInsert (see page 104)	Provides the ability to directly access the SQL object encapsulating the InsertSQL (see page 100) statement.
	QModify (see page 104)	Provides the ability to directly access the SQL object encapsulating the ModifySQL (see page 101) statement.
	QRefresh (see page 104)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL (see page 105) statement.
	QSelect (see page 104)	Provides the ability to directly access the SQL object encapsulating the SelectSQL (see page 105) statement.
	RefreshSQL (see page 105)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.
	SelectSQL (see page 105)	Provides the ability to directly access the SQL object encapsulating the SelectSQL statement.
	SelectStmtHandle (see page 105)	Gets the handle for the SELECT statement.
	SQLParams (see page 106)	Specifies the parameters to use for any parameterized queries.
	StatementType (see page 106)	Returns the statement type of the QSelect (see page 104) query.
	UpdateMode (see page 107)	Indicates how records are located when updating records.

Public Methods

	Name	Description
	AddIndex (see page 606)	Creates a new index for the table.
	ApplyUpdates (see page 69)	Writes a dataset's pending cached updates to the database.
	BookmarkValid (see page 70)	Indicates the validity of a specified bookmark.

	CachedUpdateStatus (see page 70)	Returns the status of the cached updates.
	CancelUpdates (see page 70)	Clears all pending cached updates from the cache.
V	CompareBookmarks (see page 71)	Compares two bookmarks.
V	Create (see page 605)	Creates an instance of a table component.
V	CreateBlobStream (see page 72)	Returns a TIBDSBlobStream (see page 138) object for reading or writing the data in a specified blob field.
	CreateTable (see page 606)	Builds a new table using new structure information.
	Current (see page 73)	Returns the descriptor area that holds information about the current query.
	DeleteIndex (see page 607)	Deletes an index for the table.
	DeleteTable (see page 608)	Deletes an existing database table.
V	Destroy (see page 605)	Destroys the instance of a table.
	EmptyTable (see page 608)	Deletes all records from the table.
	FetchAll (see page 74)	Retrieves all records from the current cursor position to the end of the file and stores them locally.
V	GetCurrentRecord (see page 76)	Retrieves the current record into a buffer.
V	GetDetailLinkFields (see page 609)	Lists the field components that link this dataset as a detail of a master dataset.
V	GetFieldData (see page 76)	This is the overview for the GetFieldData method overload.
	GetIndexNames (see page 609)	Retrieves a list of available indexes for a table.
	GotoCurrent (see page 610)	Synchronizes the current record for this table with the current record of a specified table component.
V	IsSequenced (see page 83)	IsSequenced indicates whether sequence numbers are available for database records.
V	Locate (see page 611)	Searches the table for a specified record and makes that record the current record.
	LocateNext (see page 85)	Searches the dataset for the record after a specified record and makes that record the current record.
V	Lookup (see page 85)	Retrieves field values from a record that matches specified search values.
V	Post (see page 86)	Writes a modified record to the database server.
	RecordModified (see page 89)	Sets the record to modified or unmodified.
	RevertRecord (see page 89)	Restores the current record in the dataset to an unmodified state when cached updates are enabled.
	SQLType (see page 91)	Returns the type of query to be executed.
	Undelete (see page 92)	Restores a record deleted from the dataset.
V	UpdateStatus (see page 92)	Reports the update status for the current record.

Public Properties

	Name	Description
R	CurrentDBKey (see page 616)	Returns the DBKey for the current row in the table.
R	DBHandle (see page 99)	Specifies the database handle for the dataset.
R	Exists (see page 616)	Indicates whether the underlying database table exists.
R	IndexFieldCount (see page 618)	Indicates the number of fields that comprise the current key.
	IndexFields (see page 619)	Retrieves or sets a field for an index.
	LiveMode (see page 619)	Indicates what types of edits, if any, can be made to the data in the dataset.
R	Plan (see page 103)	Indicates the PLAN portion of the dataset's SELECT query.
R	RowsAffected (see page 105)	Indicates the number of rows operated on by the latest query execution.
R	TableNames (see page 622)	Returns a list of table names.
R	TRHandle (see page 106)	Specifies the transaction handle for the dataset.
	UpdateRecordTypes (see page 108)	Specifies the type of records visible in a dataset when cached updates are enabled.
R	UpdatesPending (see page 108)	Indicates whether the cached updates buffer contains records that are not yet applied.

Published Events

	Name	Description
	AfterDatabaseDisconnect (see page 623)	Occurs immediately after the connection to the associated database is closed.
	AfterTransactionEnd (see page 623)	Occurs immediately after a transaction that includes this dataset's query ends.
	BeforeDatabaseDisconnect (see page 624)	Occurs immediately before the connection to the associated database is closed.

	BeforeTransactionEnd (see page 624)	Occurs immediately before a transaction that includes this dataset's query ends.
	DatabaseFree (see page 624)	Occurs when the database associated with this dataset is freed.
	OnUpdateError (see page 110)	Occurs if an exception is generated when cached updates are applied to a database.
	OnUpdateRecord (see page 111)	Occurs when cached updates are applied to a record.
	TransactionFree (see page 624)	Occurs when the transaction that includes this dataset's query is freed.

Published Properties

	Name	Description
	Active (see page 615)	This is Active, a member of class TIBTable.
	AfterCancel (see page 94)	This is AfterCancel, a member of class TIBCustomDataSet.
	AfterClose (see page 94)	This is AfterClose, a member of class TIBCustomDataSet.
	AfterDelete (see page 94)	This is AfterDelete, a member of class TIBCustomDataSet.
	AfterEdit (see page 94)	This is AfterEdit, a member of class TIBCustomDataSet.
	AfterInsert (see page 95)	This is AfterInsert, a member of class TIBCustomDataSet.
	AfterOpen (see page 95)	This is AfterOpen, a member of class TIBCustomDataSet.
	AfterPost (see page 95)	This is AfterPost, a member of class TIBCustomDataSet.
	AfterRefresh (see page 95)	This is AfterRefresh, a member of class TIBCustomDataSet.
	AfterScroll (see page 95)	This is AfterScroll, a member of class TIBCustomDataSet.
	AutoCalcFields (see page 96)	This is AutoCalcFields, a member of class TIBCustomDataSet.
	BeforeCancel (see page 96)	This is BeforeCancel, a member of class TIBCustomDataSet.
	BeforeClose (see page 96)	This is BeforeClose, a member of class TIBCustomDataSet.
	BeforeDelete (see page 96)	This is BeforeDelete, a member of class TIBCustomDataSet.
	BeforeEdit (see page 97)	This is BeforeEdit, a member of class TIBCustomDataSet.
	BeforeInsert (see page 97)	This is BeforeInsert, a member of class TIBCustomDataSet.
	BeforeOpen (see page 97)	This is BeforeOpen, a member of class TIBCustomDataSet.
	BeforePost (see page 97)	This is BeforePost, a member of class TIBCustomDataSet.
	BeforeRefresh (see page 97)	This is BeforeRefresh, a member of class TIBCustomDataSet.
	BeforeScroll (see page 98)	This is BeforeScroll, a member of class TIBCustomDataSet.
	BufferChunks (see page 615)	Specifies the dataset buffer chunk size.
	CachedUpdates (see page 615)	Specifies whether cached updates are enabled for a dataset.
	Constraints (see page 616)	This is Constraints, a member of class TIBTable.
	Database (see page 99)	Identifies the database component for which this dataset represents one or more tables.
	DefaultIndex (see page 616)	Specifies if the data in the table should be ordered on a default index when opened.
	FieldDefs (see page 617)	This is FieldDefs, a member of class TIBTable.
	Filter (see page 617)	Specifies rows in a dataset that meet the filter conditions.
	Filtered (see page 617)	Specifies whether or not filtering is active for a table.
	ForcedRefresh (see page 99)	Specifies whether the dataset should refresh its data every time it posts a record.
	IndexDefs (see page 618)	Contains information about the indexes for a table.
	IndexFieldNames (see page 618)	Lists the columns to use as an index for a table.
	IndexName (see page 619)	Identifies an index for the table.
	MasterFields (see page 620)	Specifies one or more fields in a master table to link with corresponding fields in this table in order to establish a master-detail relationship between the tables.
	MasterSource (see page 620)	Specifies the name of the data source for a dataset to use as a master table in establishing a detail-master relationship between this table and another one.
	ObjectView (see page 101)	This is ObjectView, a member of class TIBCustomDataSet.
	OnCalcFields (see page 101)	This is OnCalcFields, a member of class TIBCustomDataSet.
	OnDeleteError (see page 102)	This is OnDeleteError, a member of class TIBCustomDataSet.
	OnEditError (see page 102)	This is OnEditError, a member of class TIBCustomDataSet.
	OnFilterRecord (see page 620)	This is OnFilterRecord, a member of class TIBTable.
	OnNewRecord (see page 102)	This is OnNewRecord, a member of class TIBCustomDataSet.
	OnPostError (see page 102)	This is OnPostError, a member of class TIBCustomDataSet.
	ReadOnly (see page 621)	Specifies whether a table is read-only for this application.

	StoreDefs (see page 621)	Indicates whether the table's field and index definitions persist with the data module or form.
	TableName (see page 621)	Indicates the name of the database table or view that this component encapsulates.
	TableTypes (see page 622)	Sets the types of relations displayed in the TableName (see page 621) drop-down list.
	Transaction (see page 106)	Identifies the transaction under which the query executes.
	UniDirectional (see page 622)	Determines whether bidirectional cursors are enabled for the dataset.
	UpdateObject (see page 623)	Specifies the update object component used to update a read-only result set.

16.1.1.2 TIBTable.Create Constructor

Creates an instance of a table component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBTable(TComponent * AOwner);
```

Description

Call Create to instantiate a table declared in an application if it was not placed on a form at design time. Create calls its inherited Create constructor, creates an empty index definitions list, creates an empty data link, and creates an empty list of index files.

16.1.1.3 TIBTable.Destroy Destructor

Destroys the instance of a table.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBTable();
```

Description

Do not call Destroy directly. Instead call Free to verify that the table is not already freed before calling Destroy. Destroy frees the index files list for the table, frees its data link, frees its index definitions, and then calls its inherited Destroy destructor.

16.1.1.4 TIBTable Methods

The methods of the TIBTable class are listed here.

Protected Methods

	Name	Description
	DataEvent (see page 607)	This is DataEvent, a member of class TIBTable.
	DefChanged (see page 607)	This is DefChanged, a member of class TIBTable.
	DoOnNewRecord (see page 608)	This is DoOnNewRecord, a member of class TIBTable.
	GetCanModify (see page 608)	This is GetCanModify, a member of class TIBTable.
	GetDataSource (see page 609)	This is GetDataSource, a member of class TIBTable.
	GetIndexParams (see page 609)	This is GetIndexParams, a member of class TIBTable.
	InitFieldDefs (see page 610)	This is InitFieldDefs, a member of class TIBTable.
	InternalClose (see page 610)	This is InternalClose, a member of class TIBTable.

	InternalOpen (see page 610)	This is InternalOpen, a member of class TIBTable.
	InternalRefresh (see page 611)	This is InternalRefresh, a member of class TIBTable.
	InternalRefreshRow (see page 611)	This is InternalRefreshRow, a member of class TIBTable.
	PSGetDefaultOrder (see page 612)	This is PSGetDefaultOrder, a member of class TIBTable.
	PSGetIndexDefs (see page 612)	This is PSGetIndexDefs, a member of class TIBTable.
	PSGetKeyFields (see page 612)	This is PSGetKeyFields, a member of class TIBTable.
	PSGetTableName (see page 612)	This is PSGetTableName, a member of class TIBTable.
	PSSetCommandText (see page 613)	This is PSSetCommandText, a member of class TIBTable.
	PSSetParams (see page 613)	This is PSSetParams, a member of class TIBTable.
	SetFiltered (see page 613)	This is SetFiltered, a member of class TIBTable.
	SetFilterOptions (see page 613)	This is SetFilterOptions, a member of class TIBTable.
	SetFilterText (see page 614)	This is SetFilterText, a member of class TIBTable.
	UpdateIndexDefs (see page 614)	This is UpdateIndexDefs, a member of class TIBTable.

Public Methods

	Name	Description
	AddIndex (see page 606)	Creates a new index for the table.
	CreateTable (see page 606)	Builds a new table using new structure information.
	DeleteIndex (see page 607)	Deletes an index for the table.
	DeleteTable (see page 608)	Deletes an existing database table.
	EmptyTable (see page 608)	Deletes all records from the table.
	GetDetailLinkFields (see page 609)	Lists the field components that link this dataset as a detail of a master dataset.
	GetIndexNames (see page 609)	Retrieves a list of available indexes for a table.
	GotoCurrent (see page 610)	Synchronizes the current record for this table with the current record of a specified table component.
	Locate (see page 611)	Searches the table for a specified record and makes that record the current record.

16.1.1.4.1 TIBTable.AddIndex Method

Creates a new index for the table.

Pascal

```
procedure AddIndex(const Name: string; const Fields: string; Options: TIndexOptions; const
DescFields: string = '');
```

C++

```
__fastcall AddIndex(const UnicodeString Name, const UnicodeString Fields, TIndexOptions
Options, const UnicodeString DescFields = '');
```

Description

Call AddIndex to create a new index for the table associated with a dataset. The index created with this procedure is added to the database table underlying the dataset. Name is the name of the new index. Fields is a semicolon-delimited list of the fields to include in the index. Options is a potentially restricted set attributes for the index. It can specify that an index

- Represents the primary index for a dataset. (ixPrimary)
- Contains no duplicate values (ixUnique).
- Sorts records in descending order (isDescending).

Warning: Attempting to create an index using options that are not applicable to the table raises an exception.

16.1.1.4.2 TIBTable.CreateTable Method

Builds a new table using new structure information.

Pascal

```
procedure CreateTable;
```

C++

```
__fastcall CreateTable();
```

Description

Call CreateTable at runtime to create a table using this dataset's current definition. CreateTable overwrites an existing table's structure and data; to avoid overwriting an existing table, check Exists (see page 616) before calling CreateTable.

If the FieldDefs (see page 617) property contains values, these values are used to create field definitions. Otherwise the Fields property is used. One or both of these properties must contain values in order to recreate a dataset.

If the IndexDefs (see page 618) property contain values, these values are used to create index definitions for the dataset.

16.1.1.4.3 TIBTable.DataEvent Method

Pascal

```
procedure DataEvent(Event: TDataEvent; Info: Longint); override;
```

C++

```
virtual __fastcall DataEvent(TDataEvent Event, Longint Info);
```

Description

This is DataEvent, a member of class TIBTable.

16.1.1.4.4 TIBTable.DefChanged Method

Pascal

```
procedure DefChanged(Sender: TObject); override;
```

C++

```
virtual __fastcall DefChanged(TObject * Sender);
```

Description

This is DefChanged, a member of class TIBTable.

16.1.1.4.5 TIBTable.DeleteIndex Method

Deletes an index for the table.

Pascal

```
procedure DeleteIndex(const Name: string);
```

C++

```
__fastcall DeleteIndex(const UnicodeString Name);
```

Description

Call DeleteIndex to remove an index for a table. Name is the name of the index to delete. DeleteIndex cannot remove an index used by a constraint.

16.1.1.4.6 TIBTable.DeleteTable Method

Deletes an existing database table.

Pascal

```
procedure DeleteTable;
```

C++

```
__fastcall DeleteTable();
```

Description

Call DeleteTable to delete an existing database table associated with the table component through its Database and TableName (see page 621) properties. A table must be closed before it can be deleted.

Warning: Deleting a table erases any data the table contains and destroys the table's structure information.

16.1.1.4.7 TIBTable.DoOnNewRecord Method

Pascal

```
procedure DoOnNewRecord; override;
```

C++

```
virtual __fastcall DoOnNewRecord();
```

Description

This is DoOnNewRecord, a member of class TIBTable.

16.1.1.4.8 TIBTable.EmptyTable Method

Deletes all records from the table.

Pascal

```
procedure EmptyTable;
```

C++

```
__fastcall EmptyTable();
```

Description

The EmptyTable method deletes all records from the database table specified by the Database and TableName (see page 621) properties.

Note: Deletion of records can fail if the user lacks sufficient privileges to perform the delete operation.

16.1.1.4.9 TIBTable.GetCanModify Method

Pascal

```
function GetCanModify: Boolean; override;
```

C++

```
virtual __fastcall Boolean GetCanModify();
```

Description

This is GetCanModify, a member of class TIBTable.

16.1.1.4.10 TIBTable.GetDataSource Method

Pascal

```
function GetDataSource: TDataSource; override;
```

C++

```
virtual __fastcall TDataSource GetDataSource();
```

Description

This is GetDataSource, a member of class TIBTable.

16.1.1.4.11 TIBTable.GetDetailLinkFields Method

Lists the field components that link this dataset as a detail of a master dataset.

Pascal

```
procedure GetDetailLinkFields(MasterFields: TList; DetailFields: TList); override;
```

C++

```
virtual __fastcall GetDetailLinkFields(TList MasterFields, TList DetailFields);
```

Description

GetDetailLinkFields fills two lists of TFields that define a master-detail relationship between this table and another (master) dataset. The MasterFields (see page 620) list is filled with fields from the master table whose values must equal the values of the fields in the DetailFields list. The DetailFields list is filled with fields from the calling dataset.

16.1.1.4.12 TIBTable.GetIndexNames Method

Retrieves a list of available indexes for a table.

Pascal

```
procedure GetIndexNames(List: TStrings);
```

C++

```
__fastcall GetIndexNames(TStrings List);
```

Description

Call GetIndexNames to retrieve a list of all available indexes for a table. List is a string list object, created and maintained by the application, into which to retrieve the index names.

See Also

Obtaining Information About Indexes

16.1.1.4.13 TIBTable.GetIndexParams Method

Pascal

```
procedure GetIndexParams(const IndexName: string; FieldsIndex: Boolean; var IndexedName:
string);
```

C++

```
__fastcall GetIndexParams(const UnicodeString IndexName, Boolean FieldsIndex, UnicodeString
IndexedName);
```

Description

This is GetIndexParams, a member of class TIBTable.

16.1.1.4.14 TIBTable.GotoCurrent Method

Synchronizes the current record for this table with the current record of a specified table component.

Pascal

```
procedure GotoCurrent(Table: TIBTable);
```

C++

```
__fastcall GotoCurrent(TIBTable Table);
```

Description

Call GotoCurrent to synchronize the cursor position for this table based on the cursor position in another dataset that uses a different data source component, but which is connected to the same underlying database table. Table is the name of the table component whose cursor position to use for synchronizing.

Note: This procedure works only for table components that have the same Database and TableName (see page 621) properties. Otherwise an exception is raised.

GotoCurrent is mainly for use in applications that have two table components that are linked to the same underlying database table through different data source components. It enables an application to ensure that separate views of the data appear to be linked.

16.1.1.4.15 TIBTable.InitFieldDefs Method

Pascal

```
procedure InitFieldDefs; override;
```

C++

```
virtual __fastcall InitFieldDefs();
```

Description

This is InitFieldDefs, a member of class TIBTable.

16.1.1.4.16 TIBTable.InternalClose Method

Pascal

```
procedure InternalClose; override;
```

C++

```
virtual __fastcall InternalClose();
```

Description

This is InternalClose, a member of class TIBTable.

16.1.1.4.17 TIBTable.InternalOpen Method

Pascal

```
procedure InternalOpen; override;
```

C++

```
virtual __fastcall InternalOpen();
```

Description

This is InternalOpen, a member of class TIBTable.

16.1.1.4.18 TIBTable.InternalRefresh Method

Pascal

```
procedure InternalRefresh; override;
```

C++

```
virtual __fastcall InternalRefresh();
```

Description

This is InternalRefresh, a member of class TIBTable.

16.1.1.4.19 TIBTable.InternalRefreshRow Method

Pascal

```
procedure InternalRefreshRow; override;
```

C++

```
virtual __fastcall InternalRefreshRow();
```

Description

This is InternalRefreshRow, a member of class TIBTable.

16.1.1.4.20 TIBTable.Locate Method

Searches the table for a specified record and makes that record the current record.

Pascal

```
function Locate(const KeyFields: string; const KeyValues: Variant; Options: TLocateOptions):
Boolean; override;
```

C++

```
virtual __fastcall Boolean Locate(const UnicodeString KeyFields, const Variant KeyValues,
TLocateOptions Options);
```

Description

Call Locate to search a dataset for a specific record and position the cursor on it.

KeyFields is a string containing a semicolon-delimited list of field names on which to search.

KeyValues is a variant that specifies the values to match in the key fields. If KeyFields lists a single field, KeyValues specifies the value for that field on the desired record. To specify multiple search values, pass a variant array as KeyValues, or construct a variant array on the fly using the VarArrayOf routine. For example:

```
        with CustTable do
    Locate('Company;Contact;Phone', VarArrayOf(['Sight Diver', 'P', '408-431-1000']),
[loPartialKey]);
       TLocateOptions Opts;
    Opts.Clear();
    Opts << loPartialKey;
```

```
Variant locvalues[3];
locvalues[0] = Variant("Sight Diver");
locvalues[1] = Variant("P");
locvalues[2] = Variant("408-431-1000");
CustTable->Locate("Company;Contact;Phone", VarArrayOf(locvalues, 2), Opts);
```

Options is a set that optionally specifies additional search latitude when searching on string fields. If Options contains the loCaseInsensitive setting, then Locate ignores case when matching fields. If Options contains the loPartialKey setting, then Locate allows partial-string matching on strings in KeyValues. If Options is an empty set, or if KeyFields does not include any string fields, Options is ignored.

Locate returns true if it finds a matching record, and makes that record the current one. Otherwise Locate returns false.

Locate uses the fastest possible method to locate matching records. If the search fields in KeyFields are indexed and the index is compatible with the specified search options, Locate uses the index. Otherwise Locate creates a filter for the search.

16.1.1.4.21 TIBTable.PSGetDefaultOrder Method

Pascal

```
function PSGetDefaultOrder: TIndexDef; override;
```

C++

```
virtual __fastcall TIndexDef PSGetDefaultOrder();
```

Description

This is PSGetDefaultOrder, a member of class TIBTable.

16.1.1.4.22 TIBTable.PSGetIndexDefs Method

Pascal

```
function PSGetIndexDefs(IndexTypes: TIndexOptions): TIndexDefs; override;
```

C++

```
virtual __fastcall TIndexDefs PSGetIndexDefs(TIndexOptions IndexTypes);
```

Description

This is PSGetIndexDefs, a member of class TIBTable.

16.1.1.4.23 TIBTable.PSGetKeyFields Method

Pascal

```
function PSGetKeyFields: string; override;
```

C++

```
virtual __fastcall UnicodeString PSGetKeyFields();
```

Description

This is PSGetKeyFields, a member of class TIBTable.

16.1.1.4.24 TIBTable.PSGetTableName Method

Pascal

```
function PSGetTableName: string; override;
```

C++

```
virtual __fastcall UnicodeString PSGetTableName();
```

Description

This is PSGetTableName, a member of class TIBTable.

16.1.1.4.25 TIBTable.PSSetCommandText Method

Pascal

```
procedure PSSetCommandText(const CommandText: string); override;
```

C++

```
virtual __fastcall PSSetCommandText(const UnicodeString CommandText);
```

Description

This is PSSetCommandText, a member of class TIBTable.

16.1.1.4.26 TIBTable.PSSetParams Method

Pascal

```
procedure PSSetParams(AParams: TParams); override;
```

C++

```
virtual __fastcall PSSetParams(TParams AParams);
```

Description

This is PSSetParams, a member of class TIBTable.

16.1.1.4.27 TIBTable.SetFiltered Method

Pascal

```
procedure SetFiltered(Value: Boolean); override;
```

C++

```
virtual __fastcall SetFiltered(Boolean Value);
```

Description

This is SetFiltered, a member of class TIBTable.

16.1.1.4.28 TIBTable.SetFilterOptions Method

Pascal

```
procedure SetFilterOptions(Value: TFilterOptions); override;
```

C++

```
virtual __fastcall SetFilterOptions(TFilterOptions Value);
```

Description

This is SetFilterOptions, a member of class TIBTable.

16.1.1.4.29 TIBTable.SetFilterText Method

Pascal

```
procedure SetFilterText(const Value: string); override;
```

C++

```
virtual __fastcall SetFilterText(const UnicodeString Value);
```

Description

This is SetFilterText, a member of class TIBTable.

16.1.1.4.30 TIBTable.UpdateIndexDefs Method

Pascal

```
procedure UpdateIndexDefs; override;
```

C++

```
virtual __fastcall UpdateIndexDefs();
```

Description

This is UpdateIndexDefs, a member of class TIBTable.

16.1.1.5 TIBTable Properties

The properties of the TIBTable class are listed here.

Public Properties

	Name	Description
R	CurrentDBKey (see page 616)	Returns the DBKey for the current row in the table.
R	Exists (see page 616)	Indicates whether the underlying database table exists.
R	IndexFieldCount (see page 618)	Indicates the number of fields that comprise the current key.
	IndexFields (see page 619)	Retrieves or sets a field for an index.
	LiveMode (see page 619)	Indicates what types of edits, if any, can be made to the data in the dataset.
R	TableNames (see page 622)	Returns a list of table names.

Published Properties

	Name	Description
	Active (see page 615)	This is Active, a member of class TIBTable.
	BufferChunks (see page 615)	Specifies the dataset buffer chunk size.
	CachedUpdates (see page 615)	Specifies whether cached updates are enabled for a dataset.
	Constraints (see page 616)	This is Constraints, a member of class TIBTable.
	DefaultIndex (see page 616)	Specifies if the data in the table should be ordered on a default index when opened.
	FieldDefs (see page 617)	This is FieldDefs, a member of class TIBTable.
	Filter (see page 617)	Specifies rows in a dataset that meet the filter conditions.
	Filtered (see page 617)	Specifies whether or not filtering is active for a table.
	IndexDefs (see page 618)	Contains information about the indexes for a table.
	IndexFieldNames (see page 618)	Lists the columns to use as an index for a table.
	IndexName (see page 619)	Identifies an index for the table.
	MasterFields (see page 620)	Specifies one or more fields in a master table to link with corresponding fields in this table in order to establish a master-detail relationship between the tables.

	MasterSource (see page 620)	Specifies the name of the data source for a dataset to use as a master table in establishing a detail-master relationship between this table and another one.
	OnFilterRecord (see page 620)	This is OnFilterRecord, a member of class TIBTable.
	ReadOnly (see page 621)	Specifies whether a table is read-only for this application.
	StoreDefs (see page 621)	Indicates whether the table's field and index definitions persist with the data module or form.
	TableName (see page 621)	Indicates the name of the database table or view that this component encapsulates.
	TableTypes (see page 622)	Sets the types of relations displayed in the TableName (see page 621) drop-down list.
	UniDirectional (see page 622)	Determines whether bidirectional cursors are enabled for the dataset.
	UpdateObject (see page 623)	Specifies the update object component used to update a read-only result set.

16.1.1.5.1 TIBTable.Active Property

Pascal

```
property Active;
```

C++

```
__property Active;
```

Description

This is Active, a member of class TIBTable.

16.1.1.5.2 TIBTable.BufferChunks Property

Specifies the dataset buffer chunk size.

Pascal

```
property BufferChunks: Integer;
```

C++

```
__property int BufferChunks;
```

Description

Use BufferChunks to get or set the number of records for which the dataset allocates buffer space at any time. When the dataset's buffer is full, trying to fetch an additional record causes the dataset to reallocate the buffer so that it has enough memory to hold an additional BufferChunks records.

Note: When UniDirectional is true, BufferChunks has no meaning.

16.1.1.5.3 TIBTable.CachedUpdates Property

Specifies whether cached updates are enabled for a dataset.

Pascal

```
property CachedUpdates: Boolean;
```

C++

```
__property Boolean CachedUpdates;
```

Description

CachedUpdates enables or disables the use of cached updates for a dataset. If CachedUpdates is true, cached updates are enabled. If CachedUpdates is false, cached updates are disabled.

When cached updates are enabled, updates to a dataset (such as posting changes, inserting new records, or deleting records), are stored in an internal cache on the client machine instead of being written directly to the dataset's underlying database tables. When changes are complete, an application writes all cached changes to the database in the context of a single transaction.

Note: **Note**: Instead of using cached updates, applications can obtain the same benefits with greater control by using a client dataset.

16.1.1.5.4 TIBTable.Constraints Property

Pascal

```
property Constraints;
```

C++

```
__property Constraints;
```

Description

This is Constraints, a member of class TIBTable.

16.1.1.5.5 TIBTable.CurrentDBKey Property

Returns the DBKey for the current row in the table.

Pascal

```
property CurrentDBKey: TIBDBKey;
```

C++

```
__property TIBDBKey CurrentDBKey;
```

Description

Use CurrentDBKey to return the DBKey for the current row in the table. A DBKey is a unique row identifier for the duration of the current transaction. This property is primarily for internal use.

16.1.1.5.6 TIBTable.DefaultIndex Property

Specifies if the data in the table should be ordered on a default index when opened.

Pascal

```
property DefaultIndex: Boolean;
```

C++

```
__property Boolean DefaultIndex;
```

Description

When this property is set to false, an ORDER BY clause is not used when opening a table on the server. When DefaultIndex is true, the data is ordered based on the primary key or a unique index when opening the table. DefaultIndex defaults to true.

16.1.1.5.7 TIBTable.Exists Property

Indicates whether the underlying database table exists.

Pascal

```
property Exists: Boolean;
```

C++

```
__property Boolean Exists;
```

Description

Read Exists at runtime to determine whether a database table exists. If the table does not exist, you can create a table from the field definitions and index definitions using the CreateTable (see page 606) method. This property is read-only.

16.1.1.5.8 TIBTable.FieldDefs Property

Pascal

```
property FieldDefs;
```

C++

```
__property FieldDefs;
```

Description

This is FieldDefs, a member of class TIBTable.

16.1.1.5.9 TIBTable.Filter Property

Specifies rows in a dataset that meet the filter conditions.

Pascal

```
property Filter;
```

C++

```
__property Filter;
```

Description

Use the Filter property to restrict the rows in the dataset to those that meet the filter conditions. The syntax for Filter is the same as the search condition that appears after a WHERE clause in a Select statement. For example, to view only those records where the value in the Country field contains 'France' or 'Fiji':

Country = 'France' or Country = 'Fiji'

You can use standard SQL wildcards such as percent (%) and underscore (_) in the condition when you use the LIKE operator. The following filter condition retrieves all Countries beginning with 'F':

```
Country LIKE 'F%'
```

To view rows that have a NULL value in the Country column and Contact_Name is not NULL, use the IS operator:

```
Country is NULL and Contact_Name is not NULL
```

You can also use complex expressions in filter clauses. The following example retrieves rows containing Country values that use Francs as currency.

```
Country IN (SELECT Country from Country where Currency = '_Franc')
```

16.1.1.5.10 TIBTable.Filtered Property

Specifies whether or not filtering is active for a table.

Pascal

```
property Filtered;
```

C++

```
__property Filtered;
```

Description

Check Filtered to determine whether or not dataset filtering is in effect. If Filtered is true, then filtering is active. To apply filter conditions specified in the Filter (see page 617) property or the OnFilterRecord (see page 620) event handler, set Filtered to true.

16.1.1.5.11 TIBTable.IndexDefs Property

Contains information about the indexes for a table.

Pascal

```
property IndexDefs: TIndexDefs;
```

C++

```
__property TIndexDefs IndexDefs;
```

Description

IndexDefs is an array of index items, each of which describes an available index for the table. Ordinarily an application accesses or specifies indexes through the IndexFieldNames (see page 618) or IndexFields (see page 619) properties. If IndexDefs is updated or manually edited, the StoreDefs (see page 621) property becomes true.

Note: The index items in IndexDefs may not always reflect the current indexes available for a table. Before examining IndexDefs, call its Update method to refresh the item list.

16.1.1.5.12 TIBTable.IndexFieldCount Property

Indicates the number of fields that comprise the current key.

Pascal

```
property IndexFieldCount: Integer;
```

C++

```
__property int IndexFieldCount;
```

Description

Examine IndexFieldCount to determine the number of fields that comprise the current key. For indexes based on a single column, IndexFieldCount returns 1. For multi-column indexes, IndexFieldCount indicates the number of fields upon which the index is based.

16.1.1.5.13 TIBTable.IndexFieldNames Property

Lists the columns to use as an index for a table.

Pascal

```
property IndexFieldNames: string;
```

C++

```
__property UnicodeString IndexFieldNames;
```

Description

Use IndexFieldNames as an alternative method of specifying the index to use for a table. In IndexFieldNames specify the name

of each column to use as an index for a table. Ordering of column names is significant. Separate names with semicolon.

Note: The IndexFieldNames and IndexName (see page 619) properties are mutually exclusive. Setting one clears the other.

See Also

Creating an Index with IndexFieldNames

16.1.1.5.14 TIBTable.IndexFields Property

Retrieves or sets a field for an index.

Pascal

```
property IndexFields [Index: Integer]: TField;
```

C++

```
__property TField IndexFields[int Index];
```

Description

IndexFields provides a zero-based array of field objects, each of which corresponds to a field in the current index. Index is an ordinal value indicating the position of a field in the index. The first field in the index is IndexFields[0], the second is IndexFields[1], and so on.

Note: Do not set IndexFields directly. Instead use the IndexFieldNames (see page 618) property to order datasets on the fly at runtime.

16.1.1.5.15 TIBTable.IndexName Property

Identifies an index for the table.

Pascal

```
property IndexName: string;
```

C++

```
__property UnicodeString IndexName;
```

Description

Use IndexName to specify an index for a table. If IndexName is empty, a table's sort order is based on its primary index.

If IndexName contains a valid index name, then that index is used to determine the order of records.

Note: IndexFieldNames (see page 618) and IndexName are mutually exclusive. Setting one clears the other.

16.1.1.5.16 TIBTable.LiveMode Property

Indicates what types of edits, if any, can be made to the data in the dataset.

Pascal

```
property LiveMode: TLiveModes;
```

C++

```
__property TLiveModes LiveMode;
```

Description

Read LiveMode to determine what types of edits should be allowed. LiveMode is set automatically to restrict the capabilities of the dataset to reflect the user's rights.

Note: For read-only datasets, such as TIBQuery, LiveMode is always an empty set.

16.1.1.5.17 TIBTable.MasterFields Property

Specifies one or more fields in a master table to link with corresponding fields in this table in order to establish a master-detail relationship between the tables.

Pascal

```
property MasterFields: string;
```

C++

```
__property UnicodeString MasterFields;
```

Description

Use MasterFields after setting the MasterSource (see page 620) property to specify the names of one or more fields to use in establishing a detail-master relationship between this table and the one specified in MasterSource (see page 620).

MasterFields is a string containing one or more field names in the master table. Separate field names with semicolons.

Each time the current record in the master table changes, the new values in those fields are used to select corresponding records in this table for display.

See Also

Making the Table a Detail of Another Dataset

16.1.1.5.18 TIBTable.MasterSource Property

Specifies the name of the data source for a dataset to use as a master table in establishing a detail-master relationship between this table and another one.

Pascal

```
property MasterSource: TDataSource;
```

C++

```
__property TDataSource MasterSource;
```

Description

Use MasterSource to specify the name of the data source component whose DataSet property identifies a dataset to use as a master table in establishing a detail-master relationship between this table and another one.

After setting the MasterSource property, specify which fields to use in the master table by setting the MasterFields (see page 620) property. At runtime each time the current record in the master table changes, the new values in those fields are used to select corresponding records in this table for display.

See Also

Making the Table a Detail of Another Dataset

16.1.1.5.19 TIBTable.OnFilterRecord Property

Pascal

```
property OnFilterRecord;
```

C++

```
__property OnFilterRecord;
```

Description

This is OnFilterRecord, a member of class TIBTable.

16.1.1.5.20 TIBTable.ReadOnly Property

Specifies whether a table is read-only for this application.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use the ReadOnly property to prevent users from updating, inserting, or deleting data in the table. By default, ReadOnly is false, meaning users can potentially alter a table's data.

Note: Even if ReadOnly is false, users may not be able to modify or add data to a table. Other factors, such as insufficient SQL privileges for the application or its current user may prevent successful alterations.

To guarantee that users cannot modify or add data to a table,

1. **1** Set the Active (see page 615) property to false.
2. **2** Set ReadOnly to true.

16.1.1.5.21 TIBTable.StoreDefs Property

Indicates whether the table's field and index definitions persist with the data module or form.

Pascal

```
property StoreDefs: Boolean;
```

C++

```
__property Boolean StoreDefs;
```

Description

If StoreDefs is true, the table's index and field definitions are stored with the data module or form. Setting StoreDefs to true makes the CreateTable (see page 606) method into a one-step procedure that creates fields, indexes, and validity checks at runtime.

StoreDefs is false by default. It becomes true whenever FieldDefs (see page 617) or IndexDefs (see page 618) is updated or edited manually; to prevent edited (or imported) definitions from being stored, reset StoreDefs to false.

16.1.1.5.22 TIBTable.TableName Property

Indicates the name of the database table or view that this component encapsulates.

Pascal

```
property TableName: String;
```

C++

```
__property UnicodeString TableName;
```

Description

Use TableName to specify the name of the database relation this component encapsulates. To set TableName to a meaningful value, the Database property should already be set. If Database is set at design time, then select a valid table name from the TableName drop-down list in the Object Inspector.

Note: To set TableName, the Active (see page 615) property must be false.

Tip: Setting TableName clears the FieldDefs (see page 617) property. Therefore, if you are using TIBTable (see page 600) to create a new table, set the value of TableName before you assign any field definitions.

16.1.1.5.23 TIBTable.TableNames Property

Returns a list of table names.

Pascal

```
property TableNames: TStrings;
```

C++

```
__property TStrings TableNames;
```

Description

The TableNames property is an internal property used to display a list of the table and view names in the database.

16.1.1.5.24 TIBTable.TableTypes Property

Sets the types of relations displayed in the TableName (see page 621) drop-down list.

Pascal

```
property TableTypes: TIBTableTypes;
```

C++

```
__property TIBTableTypes TableTypes;
```

Description

Use TableTypes to change which types of relations are displayed in the TableName (see page 621) drop-down list, in addition to user tables.

16.1.1.5.25 TIBTable.UniDirectional Property

Determines whether bidirectional cursors are enabled for the dataset.

Pascal

```
property UniDirectional: Boolean;
```

C++

```
__property Boolean UniDirectional;
```

Description

Use UniDirectional to determine whether bidirectional cursors are enabled for this dataset.

Note: Do not confuse the UniDirectional property with the IsUniDirectional property inherited from TDataSet. IsUniDirectional indicates whether the dataset is a dbExpress dataset, which, in addition to using a unidirectional cursor, does not buffer records.

UniDirectional is only concerned with whether the cursor permits backward or random navigation.

16.1.1.5.26 TIBTable.UpdateObject Property

Specifies the update object component used to update a read-only result set.

Pascal

```
property UpdateObject: TIBDataSetUpdateObject;
```

C++

```
__property TIBDataSetUpdateObject UpdateObject;
```

Description

Use UpdateObject to specify the TIBDataSetUpdateObject component to use in an application that must be able to update a read-only result set.

In some cases, such as when using TIBQuery, the dataset does not automatically update records. In these cases, UpdateObject can be used to specify a TIBUpdateSQL component that performs updates as a separate transaction that is transparent to the application.

TIBDataSetUpdateObject is especially useful for complicated updates from an OnUpdateRecord event.

16.1.1.6 TIBTable Events

The events of the TIBTable class are listed here.

Published Events

	Name	Description
	AfterDatabaseDisconnect (see page 623)	Occurs immediately after the connection to the associated database is closed.
	AfterTransactionEnd (see page 623)	Occurs immediately after a transaction that includes this dataset's query ends.
	BeforeDatabaseDisconnect (see page 624)	Occurs immediately before the connection to the associated database is closed.
	BeforeTransactionEnd (see page 624)	Occurs immediately before a transaction that includes this dataset's query ends.
	DatabaseFree (see page 624)	Occurs when the database associated with this dataset is freed.
	TransactionFree (see page 624)	Occurs when the transaction that includes this dataset's query is freed.

16.1.1.6.1 TIBTable.AfterDatabaseDisconnect Event

Occurs immediately after the connection to the associated database is closed.

Pascal

```
property AfterDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterDatabaseDisconnect;
```

Description

Write an AfterDatabaseDisconnect event handler to respond when the dataset closes its connection to the database specified by the Database property.

16.1.1.6.2 TIBTable.AfterTransactionEnd Event

Occurs immediately after a transaction that includes this dataset's query ends.

Pascal

```
property AfterTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent AfterTransactionEnd;
```

Description

Write an AfterTransactionEnd event handler to respond when the transaction specified by the Transaction property ends.

16.1.1.6.3 TIBTable.BeforeDatabaseDisconnect Event

Occurs immediately before the connection to the associated database is closed.

Pascal

```
property BeforeDatabaseDisconnect: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeDatabaseDisconnect;
```

Description

Write a BeforeDatabaseDisconnect event handler to respond when the dataset is about to close its connection to the database specified by the Database property.

16.1.1.6.4 TIBTable.BeforeTransactionEnd Event

Occurs immediately before a transaction that includes this dataset's query ends.

Pascal

```
property BeforeTransactionEnd: TNotifyEvent;
```

C++

```
__property TNotifyEvent BeforeTransactionEnd;
```

Description

Write a BeforeTransactionEnd event handler to respond when the transaction specified by the Transaction property is about to end.

16.1.1.6.5 TIBTable.DatabaseFree Event

Occurs when the database associated with this dataset is freed.

Pascal

```
property DatabaseFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent DatabaseFree;
```

Description

Write a DatabaseFree event handler to respond when the database specified by the Database property is freed.

16.1.1.6.6 TIBTable.TransactionFree Event

Occurs when the transaction that includes this dataset's query is freed.

Pascal

```
property TransactionFree: TNotifyEvent;
```

C++

```
__property TNotifyEvent TransactionFree;
```

Description

Write a TransactionFree event handler to respond when the transaction specified by the Transaction property is freed.

16.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TIBTableType (see page 626)	This is record IBTable.TIBTableType.

16.2.1 IBTable.TIBTableType Enumeration

Pascal

```
TIBTableType = (
  ttSystem,
  ttView
);
```

C++

```
enum TIBTableType {
  ttSystem,
  ttView
};
```

File

IBTable

Description

This is record IBTable.TIBTableType.

16.3 Types

The following table lists types in this documentation.

Types

Name	Description
TIBTableTypes (see page 627)	TIBTableTypes and TIBTableTypes indicate the types of InterBase tables and views other than user tables.
TIndexName (see page 627)	This is type IBTable.TIndexName.

16.3.1 IBTable.TIBTableTypes Type

TIBTableTypes and TIBTableTypes indicate the types of InterBase tables and views other than user tables.

Pascal

```
TIBTableTypes = set of TIBTableType;
```

C++

```
TIBTableType TIBTableTypes;
```

File

IBTable

Description

TIBTableTypes is a set of TIBTableType (see page 626) values. The following table lists the possible values:

Value	Meaning
ttSystem	System tables and views.
ttView	User views.

16.3.2 IBTable.TIndexName Type

Pascal

```
TIndexName = String;
```

C++

```
UnicodeString TIndexName;
```

File

IBTable

Description

This is type IBTable.TIndexName.

17 IBUpdateSQL Namespace

This is namespace IBUpdateSQL.

Topics

Name	Description
Classes (see page 630)	The following table lists classes in this documentation.

Classes

	Name	Description
	TIBUpdateSQL (see page 630)	TIBUpdateSQL provides an object for updating read-only datasets when cached updates are enabled.

17.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TIBUpdateSQL (see page 630)	TIBUpdateSQL provides an object for updating read-only datasets when cached updates are enabled.

17.1.1 TIBUpdateSQL Class

TIBUpdateSQL provides an object for updating read-only datasets when cached updates are enabled.

Class Hierarchy

Pascal

```
TIBUpdateSQL = class(TIBDataSetUpdateObject);
```

C++

```
class TIBUpdateSQL : public TIBDataSetUpdateObject;
```

File

IBUpdateSQL

Description

Use a TIBUpdateSQL object to provide SQL (see page 637) statements used to update read-only datasets represented by TIBQuery (see page 376) components when cached updates are enabled. A dataset is read-only either by design or circumstance. If a dataset is read-only by design, the application itself does not provide a user interface for updating data, but may institute a programmatic scheme behind the scenes.

TIBUpdateSQL provides a mechanism for circumventing what some developers consider an SQL (see page 637)-92 limitation. It enables a developer to provide INSERT, UPDATE, DELETE, and REFRESH statements for performing separate update queries on otherwise read-only result sets in such a manner that the separate update queries are transparent to the end user.

In practical application, a TIBUpdateSQL object is placed on a data module or form, and linked to a TIBQuery (see page 376) component through that component's UpdateObject property. If the UpdateObject property points to a valid TIBUpdateSQL object, the SQL (see page 637) statements belonging to the update object are automatically applied when cached updates are applied.

See Also

Getting Started with InterBase Express

17.1.1.1 TIBUpdateSQL Members

The following tables list the members exposed by TIBUpdateSQL.

Protected Methods

	Name	Description
	GetDataSet (see page 633)	This is GetDataSet, a member of class TIBUpdateSQL.
	GetSQL (see page 633)	This is GetSQL, a member of class TIBUpdateSQL.
	SetDataSet (see page 634)	This is SetDataSet, a member of class TIBUpdateSQL.
	SQLChanged (see page 634)	This is SQLChanged, a member of class TIBUpdateSQL.

Public Methods

	Name	Description
	Apply (see page 632)	Sets the parameters for a specified SQL (see page 637) statement type, and executes the resulting statement.
	Create (see page 631)	Creates an instance of an update object.
	Destroy (see page 631)	Frees an instance of an update object.
	ExecSQL (see page 633)	Executes a specified type of SQL (see page 637) statement to perform an update for an otherwise read-only results set when cached updates is enabled.
	SetParams (see page 634)	Binds parameters in an SQL (see page 637) statement prior to statement execution.

Public Properties

	Name	Description
	DataSet (see page 635)	Identifies the dataset to which a TIBUpdateSQL (see page 630) component belongs.
	Query (see page 636)	Returns the query object used to perform a specified kind of update.
	SQL (see page 637)	Returns a specified SQL statement used when applying cached updates.

Published Properties

	Name	Description
	DeleteSQL (see page 635)	Specifies the SQL (see page 637) DELETE statement to use when applying a cached deletion of a record.
	InsertSQL (see page 636)	Specifies the SQL (see page 637) INSERT statement to use when applying a cached insertion of a record.
	ModifySQL (see page 636)	Specifies the SQL (see page 637) UPDATE statement to use when applying an update to a record and cached updates is enabled.
	RefreshSQL (see page 137)	Provides the ability to directly access the SQL object encapsulating the RefreshSQL statement.

17.1.1.2 TIBUpdateSQL.Create Constructor

Creates an instance of an update object.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TIBUpdateSQL(TComponent * AOwner);
```

Description

Call Create to instantiate an update object at run time. You do not need to call Create for update objects placed in a data module or form at design time. Delphi automatically handles these objects.

17.1.1.3 TIBUpdateSQL.Destroy Destructor

Frees an instance of an update object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIBUpdateSQL();
```

Description

Do not call Destroy directly in an application. Usually destruction of update objects is handled automatically by Delphi. If an application creates its own instance of an update object, however, the application should call Free, which verifies that the update object is not already freed before calling Destroy.

17.1.1.4 TIBUpdateSQL Methods

The methods of the TIBUpdateSQL class are listed here.

Protected Methods

	Name	Description
	GetDataSet (see page 633)	This is GetDataSet, a member of class TIBUpdateSQL.
	GetSQL (see page 633)	This is GetSQL, a member of class TIBUpdateSQL.
	SetDataSet (see page 634)	This is SetDataSet, a member of class TIBUpdateSQL.
	SQLChanged (see page 634)	This is SQLChanged, a member of class TIBUpdateSQL.

Public Methods

	Name	Description
	Apply (see page 632)	Sets the parameters for a specified SQL (see page 637) statement type, and executes the resulting statement.
	ExecSQL (see page 633)	Executes a specified type of SQL (see page 637) statement to perform an update for an otherwise read-only results set when cached updates is enabled.
	SetParams (see page 634)	Binds parameters in an SQL (see page 637) statement prior to statement execution.

17.1.1.4.1 TIBUpdateSQL.Apply Method

Sets the parameters for a specified SQL (see page 637) statement type, and executes the resulting statement.

Pascal

```
procedure Apply(UpdateKind: TUpdateKind); override;
```

C++

```
virtual __fastcall Apply(TUpdateKind UpdateKind);
```

Description

Call Apply to set parameters for an SQL (see page 637) statement and execute it to update a record. UpdateKind indicates which SQL (see page 637) statement to bind and execute, and can be one of the following values:

Value Meaning

ukModify Bind and execute the SQL (see page 637) statement in the ModifySQL (see page 636) property

ukInsert Bind and execute the SQL (see page 637) statement in the InsertSQL (see page 636) property

ukDelete Bind and execute the SQL (see page 637) statement in the DeleteSQL (see page 635) property

Apply is primarily intended for manually executing update statements from an OnUpdateRecord event handler.

Note: If an SQL (see page 637) statement does not contain parameters, it is more efficient to call ExecSQL (see page 633)

instead of Apply.

17.1.1.4.2 TIBUpdateSQL.ExecSQL Method

Executes a specified type of SQL (see page 637) statement to perform an update for an otherwise read-only results set when cached updates is enabled.

Pascal

```
procedure ExecSQL(UpdateKind: TUpdateKind);
```

C++

```
__fastcall ExecSQL(TUpdateKind UpdateKind);
```

Description

Call ExecSQL to execute the SQL (see page 637) statement necessary for updating the records belonging to a read-only result set when cached updates is enabled. UpdateKind specifies the statement to execute, and can be one of the following values:

Value Meaning

ukModify Execute the SQL (see page 637) statement used to update records in the dataset

ukInsert Execute the SQL (see page 637) statement used to insert new records into the dataset

ukDelete Execute the SQL (see page 637) statement used to delete records in the dataset.

If the statement to execute contains any parameters, an application must call SetParams (see page 634) to bind the parameters before calling ExecSQL. To determine if a statement contains parameters, examine the appropriate ModifySQL (see page 636), InsertSQL (see page 636), or DeleteSQL (see page 635) property, depending on the statement type intended for execution.

Note: To both bind parameters and execute a statement, call Apply (see page 632).

17.1.1.4.3 TIBUpdateSQL.GetDataSet Method

Pascal

```
function GetDataSet: TIBCustomDataSet; override;
```

C++

```
virtual __fastcall TIBCustomDataSet GetDataSet();
```

Description

This is GetDataSet, a member of class TIBUpdateSQL.

17.1.1.4.4 TIBUpdateSQL.GetSQL Method

Pascal

```
function GetSQL(UpdateKind: TUpdateKind): TStrings; override;
```

C++

```
virtual __fastcall TStrings GetSQL(TUpdateKind UpdateKind);
```

Description

This is GetSQL, a member of class TIBUpdateSQL.

17.1.1.4.5 TIBUpdateSQL.SetDataSet Method

Pascal

```
procedure SetDataSet(ADataSet: TIBCustomDataSet); override;
```

C++

```
virtual __fastcall SetDataSet(TIBCustomDataSet ADataSet);
```

Description

This is SetDataSet, a member of class TIBUpdateSQL.

17.1.1.4.6 TIBUpdateSQL.SetParams Method

Binds parameters in an SQL (see page 637) statement prior to statement execution.

Pascal

```
procedure SetParams(UpdateKind: TUpdateKind);
```

C++

```
__fastcall SetParams(TUpdateKind UpdateKind);
```

Description

Call SetParams to bind parameters in an SQL (see page 637) statement associated with the update object prior to executing the statement. UpdateKind indicates the type of statement for which to bind parameters, and can be one of the following values:

Value	Meaning
ukModify	Bind parameters for the SQL (see page 637) statement used to update records
ukInsert	Bind parameters for the SQL (see page 637) statement used to insert new records
ukDelete	Bind parameters for the SQL (see page 637) statement used to delete records

Parameters are indicated in an SQL (see page 637) statement by a colon. Except for the leading colon in the parameter name, the parameter name must exactly match the name of an existing field name for the dataset.

Note: Parameter names can be prefaced by the 'OLD_' indicator. If so, the old value of the field is used to perform the update instead of any updates in the cache.

17.1.1.4.7 TIBUpdateSQL.SQLChanged Method

Pascal

```
procedure SQLChanged(Sender: TObject);
```

C++

```
__fastcall SQLChanged(TObject * Sender);
```

Description

This is SQLChanged, a member of class TIBUpdateSQL.

17.1.1.5 TIBUpdateSQL Properties

The properties of the TIBUpdateSQL class are listed here.

Public Properties

	Name	Description
	DataSet (see page 635)	Identifies the dataset to which a TIBUpdateSQL (see page 630) component belongs.
R	Query (see page 636)	Returns the query object used to perform a specified kind of update.
	SQL (see page 637)	Returns a specified SQL statement used when applying cached updates.

Published Properties

	Name	Description
	DeleteSQL (see page 635)	Specifies the SQL (see page 637) DELETE statement to use when applying a cached deletion of a record.
	InsertSQL (see page 636)	Specifies the SQL (see page 637) INSERT statement to use when applying a cached insertion of a record.
	ModifySQL (see page 636)	Specifies the SQL (see page 637) UPDATE statement to use when applying an update to a record and cached updates is enabled.

17.1.1.5.1 TIBUpdateSQL.DataSet Property

Identifies the dataset to which a TIBUpdateSQL (see page 630) component belongs.

Pascal

```
property DataSet: TIBCustomDataSet;
```

C++

```
__property TIBCustomDataSet DataSet;
```

Description

At design time, setting the dataset object's UpdateObject property automatically sets the DataSet property of the specified TIBUpdateSQL (see page 630) object. An application should only need to set this property if it creates a new update component at run time.

17.1.1.5.2 TIBUpdateSQL.DeleteSQL Property

Specifies the SQL (see page 637) DELETE statement to use when applying a cached deletion of a record.

Pascal

```
property DeleteSQL: TStrings;
```

C++

```
__property TStrings DeleteSQL;
```

Description

Set DeleteSQL to the SQL (see page 637) DELETE statement to use when applying a deletion to a record. Statements can be parameterized queries. To create a DELETE statement at design time, use the UpdateSQL editor to create statements, such as:

```
delete from Employee
    where
    Emp_No = :OLD_Emp_No
```

At run time, an application can write a statement directly to this property to set or change the DELETE statement.

Note: As the example illustrates, DeleteSQL supports an extension to normal parameter binding. To retrieve the value of a field as it exists prior to application of cached updates, use the field name with the prefix 'OLD_'. This is especially useful when doing field comparisons in the WHERE clause of the statement.

17.1.1.5.3 TIBUpdateSQL.InsertSQL Property

Specifies the SQL (see page 637) INSERT statement to use when applying a cached insertion of a record.

Pascal

```
property InsertSQL: TStrings;
```

C++

```
__property TStrings InsertSQL;
```

Description

Set InsertSQL to the SQL (see page 637) INSERT statement to use when applying an insertion to a dataset. Statements can be parameterized queries. To create a INSERT statement at design time, use the UpdateSQL editor to create statements, such as:

```
insert into Country
    (Country, Currency)
    values (:Country, :Currency)
```

At run time, an application can write a statement directly to this property to set or change the INSERT statement.

17.1.1.5.4 TIBUpdateSQL.ModifySQL Property

Specifies the SQL (see page 637) UPDATE statement to use when applying an update to a record and cached updates is enabled.

Pascal

```
property ModifySQL: TStrings;
```

C++

```
__property TStrings ModifySQL;
```

Description

Set ModifySQL to the SQL (see page 637) UPDATE statement to use when applying an updated record to a dataset. Statements can be parameterized queries. To create a UPDATE statement at design time, use the UpdateSQL editor to create statements, such as:

```
update Employee
    set Last_Name = :Last_Name
    where Emp_No = :OLD_Emp_No
```

At run time, an application can write a statement directly to this property to set or change the UPDATE statement.

Note: As the example illustrates, ModifySQL supports an extension to normal parameter binding. To retrieve the value of a field as it exists prior to application of cached updates, the field name with 'OLD_'. This is especially useful when doing field comparisons in the WHERE clause of the statement.

17.1.1.5.5 TIBUpdateSQL.Query Property

Returns the query object used to perform a specified kind of update.

Pascal

```
property Query [UpdateKind: TUpdateKind]: TIBQuery;
```

C++

```
__property TIBQuery Query[TUpdateKind UpdateKind];
```

Description

Query returns the TIBQuery (see page 376) object used to perform a particular form of SQL (see page 637) update. UpdateKind specifies which query object to retrieve. UpdateKind can be one of the following:

Value Meaning

ukModify Return the query object used to execute UPDATE statements

ukInsert Return the query object used to execute INSERT statements

ukDelete Return the query object used to execute DELETE statements

Each query object executes a particular kind of SQL (see page 637) statement. The contents of the SQL (see page 637) statements executed by these objects can be accessed directly using the ModifySQL (see page 636), InsertSQL (see page 636), and DeleteSQL (see page 635) properties.

The main purpose of Query is to provide a way for an application to set the properties for an update query object or to call the query object's methods.

Note: If a particular kind of update statement is not provided, then its corresponding query object is nil (Delphi) or NULL (C++). For example, if an application does not provide an SQL (see page 637) statement for the DeleteSQL (see page 635) property, then Query[ukDelete] returns nil or NULL.

17.1.1.5.6 TIBUpdateSQL.SQL Property

Returns a specified SQL statement used when applying cached updates.

Pascal

```
property SQL [UpdateKind: TUpdateKind]: TStrings;
```

C++

```
__property TStrings SQL[TUpdateKind UpdateKind];
```

Description

Returns the SQL statement in the ModifySQL (see page 636), InsertSQL (see page 636), or DeleteSQL (see page 635) property, depending on the setting of UpdateKind. UpdateKind can be any of the following:

Value Meaning

ukModify Return the SQL statement used to update records in the dataset

ukInsert Return the SQL statement used to insert new records into the dataset

ukDelete Return the SQL statement used to delete records in the dataset

18 IBUtils Namespace

This is namespace IBUtils.

Topics

Name	Description
Functions (see page 640)	The following table lists functions in this documentation.
Constants (see page 644)	The following table lists constants in this documentation.

Constants

Name	Description
CR (see page 644)	This is constant IBUtils.CR.
CRLF (see page 644)	This is constant IBUtils.CRLF.
LF (see page 645)	This is constant IBUtils.LF.
NULL_TERMINATOR (see page 645)	This is constant IBUtils.NULL_TERMINATOR.
TAB (see page 645)	This is constant IBUtils.TAB.

Functions

	Name	Description
	AddIBParamSQLForDetail (see page 640)	This is function IBUtils.AddIBParamSQLForDetail.
	ExtractIdentifier (see page 640)	This is function IBUtils.ExtractIdentifier.
	FormatIdentifier (see page 641)	This is function IBUtils.FormatIdentifier.
	FormatIdentifierValue (see page 641)	This is function IBUtils.FormatIdentifierValue.
	Max (see page 641)	Returns the greater of two numeric values (Delphi only).
	Min (see page 642)	Returns the lesser of two numeric values (Delphi only).
	QuoteIdentifier (see page 642)	This is function IBUtils.QuoteIdentifier.
	RandomInteger (see page 642)	This is function IBUtils.RandomInteger.
	RandomString (see page 643)	This is function IBUtils.RandomString.
	StripString (see page 643)	This is function IBUtils.StripString.

18.1 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	AddIBParamSQLForDetail (see page 640)	This is function IBUtils.AddIBParamSQLForDetail.
	ExtractIdentifier (see page 640)	This is function IBUtils.ExtractIdentifier.
	FormatIdentifier (see page 641)	This is function IBUtils.FormatIdentifier.
	FormatIdentifierValue (see page 641)	This is function IBUtils.FormatIdentifierValue.
	Max (see page 641)	Returns the greater of two numeric values (Delphi only).
	Min (see page 642)	Returns the lesser of two numeric values (Delphi only).
	QuoteIdentifier (see page 642)	This is function IBUtils.QuoteIdentifier.
	RandomInteger (see page 642)	This is function IBUtils.RandomInteger.
	RandomString (see page 643)	This is function IBUtils.RandomString.
	StripString (see page 643)	This is function IBUtils.StripString.

18.1.1 IBUtils.AddIBParamSQLForDetail Function

Pascal

```
function AddIBParamSQLForDetail(Params: TParams; SQL: string; Native: Boolean; Dialect:
Integer): string;
```

C++

```
UnicodeString AddIBParamSQLForDetail(TParams Params, UnicodeString SQL, Boolean Native, int
Dialect);
```

File

IBUtils

Description

This is function IBUtils.AddIBParamSQLForDetail.

18.1.2 IBUtils.ExtractIdentifier Function

Pascal

```
function ExtractIdentifier(Dialect: Integer; Value: String): String;
```

C++

```
UnicodeString ExtractIdentifier(int Dialect, UnicodeString Value);
```

File

IBUtils

Description

This is function IBUtils.ExtractIdentifier.

18.1.3 IBUtils.FormatIdentifier Function

Pascal

```
function FormatIdentifier(Dialect: Integer; Value: String): String;
```

C++

```
UnicodeString FormatIdentifier(int Dialect, UnicodeString Value);
```

File

IBUtils

Description

This is function IBUtils.FormatIdentifier.

18.1.4 IBUtils.FormatIdentifierValue Function

Pascal

```
function FormatIdentifierValue(Dialect: Integer; Value: String): String;
```

C++

```
UnicodeString FormatIdentifierValue(int Dialect, UnicodeString Value);
```

File

IBUtils

Description

This is function IBUtils.FormatIdentifierValue.

18.1.5 IBUtils.Max Function

Returns the greater of two numeric values (Delphi only).

Pascal

```
function Max(n1: Integer; n2: Integer): Integer;
```

C++

```
int Max(int n1, int n2);
```

File

IBUtils

Description

In Delphi, call Max to compare two numeric values. Max returns the greater value of the two.

See Also

MaxValue (see page 1240)

Min (see page 1241)

MaxIntValue (see page 1239)

18.1.6 IBUtils.Min Function

Returns the lesser of two numeric values (Delphi only).

Pascal

```
function Min(n1: Integer; n2: Integer): Integer;
```

C++

```
int Min(int n1, int n2);
```

File

IBUtils

Description

Call Min to compare two numeric values in Delphi. Min returns the smaller value of the two.

See Also

MinValue (see page 1242)

Max (see page 1239)

MinIntValue (see page 1242)

18.1.7 IBUtils.QuoteIdentifier Function

Pascal

```
function QuoteIdentifier(Dialect: Integer; Value: String): String;
```

C++

```
UnicodeString QuoteIdentifier(int Dialect, UnicodeString Value);
```

File

IBUtils

Description

This is function IBUtils.QuoteIdentifier.

18.1.8 IBUtils.RandomInteger Function

Pascal

```
function RandomInteger(iLow: Integer; iHigh: Integer): Integer;
```

C++

```
int RandomInteger(int iLow, int iHigh);
```

File

IBUtils

Description

This is function IBUtils.RandomInteger.

18.1.9 IBUtils.RandomString Function

Pascal

```
function RandomString(iLength: Integer): String;
```

C++

```
UnicodeString RandomString(int iLength);
```

File

IBUtils

Description

This is function IBUtils.RandomString.

18.1.10 IBUtils.StripString Function

Pascal

```
function StripString(st: String; CharsToStrip: String): String;
```

C++

```
UnicodeString StripString(UnicodeString st, UnicodeString CharsToStrip);
```

File

IBUtils

Description

This is function IBUtils.StripString.

18.2 Constants

The following table lists constants in this documentation.

Constants

Name	Description
CR (see page 644)	This is constant IBUtils.CR.
CRLF (see page 644)	This is constant IBUtils.CRLF.
LF (see page 645)	This is constant IBUtils.LF.
NULL_TERMINATOR (see page 645)	This is constant IBUtils.NULL_TERMINATOR.
TAB (see page 645)	This is constant IBUtils.TAB.

18.2.1 IBUtils.CR Constant

Pascal

```
CR = #13;
```

C++

```
CR = #13;
```

File

IBUtils

Description

This is constant IBUtils.CR.

18.2.2 IBUtils.CRLF Constant

Pascal

```
CRLF = #13 + #10;
```

C++

```
CRLF = #13 + #10;
```

File

IBUtils

Description

This is constant IBUtils.CRLF.

18.2.3 IBUtils.LF Constant

Pascal

```
LF = #10;
```

C++

```
LF = #10;
```

File

IBUtils

Description

This is constant IBUtils.LF.

18.2.4 IBUtils.NULL_TERMINATOR Constant

Pascal

```
NULL_TERMINATOR = #0;
```

C++

```
NULL_TERMINATOR = #0;
```

File

IBUtils

Description

This is constant IBUtils.NULL_TERMINATOR.

18.2.5 IBUtils.TAB Constant

Pascal

```
TAB = #9;
```

C++

```
TAB = #9;
```

File

IBUtils

Description

This is constant IBUtils.TAB.

19 ImgList Namespace

This is namespace ImgList.

Topics

Name	Description
Classes (see page 648)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 685)	The following table lists structs, records, enums in this documentation.
Types (see page 689)	The following table lists types in this documentation.

Classes

	Name	Description
	TChangeLink (see page 648)	The TImageList component uses TChangeLink internally to notify other objects when changes are made to the image list.
	TCustomImageList (see page 651)	TCustomImageList is the base class for collections of same-sized images, each of which can be referred to by its index.

Structs, Records, Enums

	Name	Description
	TColorDepth (see page 685)	Color depth of list images.
	TDrawingStyle (see page 686)	TDrawingStyle indicates the style to use when rendering an image from an image list.
	TImageType (see page 687)	TImageType indicates whether to use a mask when rendering an image from an image list.
	TLoadResource (see page 687)	TLoadResource and TLoadResources (see page 689) determine how a resource is loaded into an image list.
	TResType (see page 688)	TResType identifies a type of graphical resource to an image list.

Types

Name	Description
TImageIndex (see page 689)	This is type ImgList.TImageIndex.
TImageListHandle (see page 689)	This is type ImgList.TImageListHandle.
TLoadResources (see page 689)	TLoadResource (see page 687) and TLoadResources determine how a resource is loaded into an image list.
TOverlay (see page 690)	TOverlay indicates the index of an overlay mask.

19.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TChangeLink (see page 648)	The TImageList component uses TChangeLink internally to notify other objects when changes are made to the image list.
	TCustomImageList (see page 651)	TCustomImageList is the base class for collections of same-sized images, each of which can be referred to by its index.

19.1.1 TChangeLink Class

The TImageList component uses TChangeLink internally to notify other objects when changes are made to the image list.

Class Hierarchy

Pascal

```
TChangeLink = class(TObject);
```

C++

```
class TChangeLink : public TObject;
```

File

ImgList

Description

A TImageList keeps a list of its TChangeLink objects. Each is associated with a particular component, such as a tree view or list view control. A component can register its change link with an image list by calling the image list's RegisterChanges method. Before destruction, the component should then call the UnRegisterChanges method.

Component writers can use the change link object to receive notification when changes are made to an image list that a control uses. Use the OnChange (see page 650) event of the change link to execute some code when the image list changes. For example, the tree view control uses a change link object to know when its image list has changed.

See Also

TImageList

TTreeView

TListView

TCustomImageList (see page 651)

19.1.1.1 TChangeLink Members

The following tables list the members exposed by TChangeLink.

Public Events

	Name	Description
	OnChange (see page 650)	Occurs when there's a change in any of the images stored in the associated image list.

Public Methods

	Name	Description
	Change (see page 649)	Generates an OnChange (see page 650) event.
	Destroy (see page 649)	Destroys the change link object and releases the memory allocated to it.

Public Properties

	Name	Description
	Sender (see page 650)	Specifies which image list component notifies the change link object of changes.

19.1.1.2 TChangeLink.Destroy Destructor

Destroys the change link object and releases the memory allocated to it.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TChangeLink();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the change link is not nil before calling Destroy.

Before freeing the memory for the change link, Destroy first calls the image list's UnRegisterChanges method to unregister itself from receiving notification when the image list changes.

See Also

System::TObject::Free

UnRegisterChanges (see page 674)

19.1.1.3 TChangeLink Methods

The methods of the TChangeLink class are listed here.

Public Methods

	Name	Description
	Change (see page 649)	Generates an OnChange (see page 650) event.

19.1.1.3.1 TChangeLink.Change Method

Generates an OnChange (see page 650) event.

Pascal

```
procedure Change; dynamic;
```

C++

```
__fastcall Change();
```

Description

Use the Change method to trigger the OnChange (see page 650) event handler of the change link object.

See Also

OnChange (see page 650)

19.1.1.4 TChangeLink Properties

The properties of the TChangeLink class are listed here.

Public Properties

	Name	Description
	Sender (see page 650)	Specifies which image list component notifies the change link object of changes.

19.1.1.4.1 TChangeLink.Sender Property

Specifies which image list component notifies the change link object of changes.

Pascal

```
property Sender: TCustomImageList;
```

C++

```
__property TCustomImageList Sender;
```

Description

Use Sender to access the image list that has changed. The TImageList automatically sets this property when a call to RegisterChanges is made.

See Also

RegisterChanges (see page 672)

19.1.1.5 TChangeLink Events

The events of the TChangeLink class are listed here.

Public Events

	Name	Description
	OnChange (see page 650)	Occurs when there's a change in any of the images stored in the associated image list.

19.1.1.5.1 TChangeLink.OnChange Event

Occurs when there's a change in any of the images stored in the associated image list.

Pascal

```
property OnChange: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnChange;
```

Description

Use the OnChange event to notify a control when the image list that it's using changes. Specify any special processing to occur at that time in the OnChange event handler.

See Also

Change (see page 649)

19.1.2 TCustomImageList Class

TCustomImageList is the base class for collections of same-sized images, each of which can be referred to by its index.

Class Hierarchy

Pascal

```
[RootDesignerSerializerAttribute('', '', False)]
TCustomImageList = class(TComponent);
```

C++

```
[RootDesignerSerializerAttribute('', '', False)]
class TCustomImageList : public TComponent;
```

File

ImgList

Description

Image lists manage large sets of icons or bitmaps. All images in an image list are contained in a single, wide bitmap in screen device format. An image list may also include a monochrome bitmap that contains masks that are combined with the images to produce a transparent effect such as is common in icons.

See Also

TPicture

TImage

19.1.2.1 TCustomImageList Members

The following tables list the members exposed by TCustomImageList.

Protected Methods

	Name	Description
	AssignTo (see page 658)	Copies the image list's properties to another image list.
	Change (see page 658)	Generates an OnChange (see page 684) event.
	DefineProperties (see page 659)	Allows bitmap data to be saved and loaded.
	DoDraw (see page 659)	Draws the specified image onto the provided Canvas in the indicated style.
	GetImages (see page 666)	Returns a specified image and its mask as bitmaps.
	HandleNeeded (see page 669)	Creates an image list handle for the image list if one doesn't already exist.
	Initialize (see page 669)	Initializes a newly allocated image list.
	ReadData (see page 672)	This is ReadData, a member of class TCustomImageList.
	WriteData (see page 675)	This is WriteData, a member of class TCustomImageList.

Public Events

	Name	Description
	OnChange (see page 684)	Occurs when the image list changes.

Public Methods

	Name	Description
	Add (see page 655)	Adds an Image and its Mask (see page 1079) to the image list object.
	AddIcon (see page 655)	Adds an icon to the image list object.
	AddImage (see page 656)	Adds single image from another image list.
	AddImages (see page 656)	Adds all images from another image list.
	AddMasked (see page 657)	Adds an image to the image list along with a color which is used to generate the transparent mask.
V	Assign (see page 657)	Copies the information from one image list to another.
	Clear (see page 658)	Clears all images from the image list.
V	Create (see page 653)	Creates an empty image list.
	CreateSize (see page 653)	Creates an empty image list of a specified size.
	Delete (see page 659)	Deletes a particular image from the image list.
V	Destroy (see page 654)	Destroys the image list and releases the memory allocated to it.
	Draw (see page 660)	This is the overview for the Draw method overload.
	DrawOverlay (see page 663)	This is the overview for the DrawOverlay method overload.
	FileLoad (see page 664)	Loads a resourced image into the image list.
	GetBitmap (see page 664)	Retrieves a specified image as a bitmap.
V	GetHotSpot (see page 665)	Returns the hot spot location associated with the current drag image.
	GetIcon (see page 665)	This is the overview for the GetIcon method overload.
	GetImageBitmap (see page 666)	Returns a handle to a bitmap containing all of the images in the image list.
	GetInstRes (see page 667)	This is the overview for the GetInstRes method overload.
	GetMaskBitmap (see page 668)	Returns a handle to a bitmap containing all masks for images in the image list.
	GetResource (see page 668)	Loads the specified bitmap, cursor, or icon resource into the image list.
	HandleAllocated (see page 669)	Indicates whether a window handle exists for the image list.
	Insert (see page 670)	Inserts a bitmap and its mask into the image list following a specified image.
	InsertIcon (see page 670)	Inserts an icon at a specified position in the image list.
	InsertMasked (see page 670)	Inserts a bitmap at a specified position, creating the mask from a specified transparent color.
	Move (see page 671)	Moves a specified image to another position.
	Overlay (see page 671)	Identifies an image in the list as an image that can be used as an overlay mask.
	RegisterChanges (see page 672)	Registers an object to be notified when changes occur in the image list.
	Replace (see page 672)	Replaces an image in the image list with a new image and mask.
	ReplaceIcon (see page 673)	Replaces an image in the image list with an icon.
	ReplaceMasked (see page 673)	Replaces an image in the image list with a new image and automatically-generated mask.
	ResInstLoad (see page 673)	Loads a resource into the image list.
	ResourceLoad (see page 674)	Loads a resource into the image list.
	SetSize (see page 674)	This is SetSize, a member of class TCustomImageList.
	UnRegisterChanges (see page 674)	Removes an object from the list of objects to be notified when the image list changes.

Public Properties

	Name	Description
	AllocBy (see page 675)	Sets the number of images by which the image list grows when it needs to make room for new images.
	BkColor (see page 676)	Determines which background color to use when drawing an image.
	BlendColor (see page 676)	Determines which foreground color to use when drawing an image.
R	Count (see page 677)	Indicates the number of images in the image list.
	DrawingStyle (see page 681)	Specifies the style to be used when the image list is drawing an image.
	Handle (see page 682)	Designates the handle for the image list.

	Height (see page 682)	Specifies the height of images within the image list.
	ImageType (see page 682)	Determines whether the image list uses the associated mask when drawing.
	Masked (see page 683)	Specifies whether the image list includes masks that can be combined with the images.
	ShareImages (see page 683)	Specifies whether the image list destroys its handle when the image list is destroyed.
	Width (see page 684)	Specifies the width of all images within the image list.

Published Properties

	Name	Description
	ColorDepth (see page 677)	Color depth of images in list.

19.1.2.2 TCustomImageList.Create Constructor

Creates an empty image list.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomImageList(TComponent * AOwner);
```

Description

Call Create to instantiate an image list at runtime. ImageLists placed on forms or data modules at design time are created automatically. Specify the owner of the new imagelist using the AOwner parameter.

The new image list contains no images and has a default size of 16 by 16 pixels.

See Also

CreateSize (see page 653)

19.1.2.3 TCustomImageList.CreateSize Constructor

Creates an empty image list of a specified size.

Pascal

```
constructor CreateSize(AWidth: Integer; AHeight: Integer);
```

C++

```
__fastcall TCustomImageList(int AWidth, int AHeight);
```

Description

Use CreateSize to create an image list object at runtime with an image size other than the default of 16 x 16 pixels. The initial size of the image list is determined by the AHeight and AWidth parameters. The height and width can be altered after the image list has been created using the Width (see page 684) and Height (see page 682) properties.

See Also

Create (see page 653)

Height (see page 682)

Width (see page 684)

19.1.2.4 TCustomImageList.Destroy Destructor

Destroys the image list and releases the memory allocated to it.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomImageList();
```

Description

Destroy first calls UnRegisterChanges (see page 674) to unregister any objects registered to receive notification when the image list changed, and frees all helper objects associated with the image list.

Destroy seldom needs to be called. However, when an image list is constructed without an owner by the Create (see page 653) or CreateSize (see page 653) method, call Free to release memory and dispose of the object. Free checks to see if the pointer is nil before calling Destroy.

See Also

System::TObject::Free

UnRegisterChanges (see page 674)

19.1.2.5 TCustomImageList Methods

The methods of the TCustomImageList class are listed here.

Protected Methods

	Name	Description
	AssignTo (see page 658)	Copies the image list's properties to another image list.
	Change (see page 658)	Generates an OnChange (see page 684) event.
	DefineProperties (see page 659)	Allows bitmap data to be saved and loaded.
	DoDraw (see page 659)	Draws the specified image onto the provided Canvas in the indicated style.
	GetImages (see page 666)	Returns a specified image and its mask as bitmaps.
	HandleNeeded (see page 669)	Creates an image list handle for the image list if one doesn't already exist.
	Initialize (see page 669)	Initializes a newly allocated image list.
	ReadData (see page 672)	This is ReadData, a member of class TCustomImageList.
	WriteData (see page 675)	This is WriteData, a member of class TCustomImageList.

Public Methods

	Name	Description
	Add (see page 655)	Adds an Image and its Mask (see page 1079) to the image list object.
	AddIcon (see page 655)	Adds an icon to the image list object.
	AddImage (see page 656)	Adds single image from another image list.
	AddImages (see page 656)	Adds all images from another image list.
	AddMasked (see page 657)	Adds an image to the image list along with a color which is used to generate the transparent mask.
	Assign (see page 657)	Copies the information from one image list to another.
	Clear (see page 658)	Clears all images from the image list.
	Delete (see page 659)	Deletes a particular image from the image list.
	Draw (see page 660)	This is the overview for the Draw method overload.
	DrawOverlay (see page 663)	This is the overview for the DrawOverlay method overload.
	FileLoad (see page 664)	Loads a resourced image into the image list.

	GetBitmap (see page 664)	Retrieves a specified image as a bitmap.
	GetHotSpot (see page 665)	Returns the hot spot location associated with the current drag image.
	GetIcon (see page 665)	This is the overview for the GetIcon method overload.
	GetImageBitmap (see page 666)	Returns a handle to a bitmap containing all of the images in the image list.
	GetInstRes (see page 667)	This is the overview for the GetInstRes method overload.
	GetMaskBitmap (see page 668)	Returns a handle to a bitmap containing all masks for images in the image list.
	GetResource (see page 668)	Loads the specified bitmap, cursor, or icon resource into the image list.
	HandleAllocated (see page 669)	Indicates whether a window handle exists for the image list.
	Insert (see page 670)	Inserts a bitmap and its mask into the image list following a specified image.
	InsertIcon (see page 670)	Inserts an icon at a specified position in the image list.
	InsertMasked (see page 670)	Inserts a bitmap at a specified position, creating the mask from a specified transparent color.
	Move (see page 671)	Moves a specified image to another position.
	Overlay (see page 671)	Identifies an image in the list as an image that can be used as an overlay mask.
	RegisterChanges (see page 672)	Registers an object to be notified when changes occur in the image list.
	Replace (see page 672)	Replaces an image in the image list with a new image and mask.
	ReplaceIcon (see page 673)	Replaces an image in the image list with an icon.
	ReplaceMasked (see page 673)	Replaces an image in the image list with a new image and automatically-generated mask.
	ResInstLoad (see page 673)	Loads a resource into the image list.
	ResourceLoad (see page 674)	Loads a resource into the image list.
	SetSize (see page 674)	This is SetSize, a member of class TCustomImageList.
	UnRegisterChanges (see page 674)	Removes an object from the list of objects to be notified when the image list changes.

19.1.2.5.1 TCustomImageList.Add Method

Adds an Image and its Mask (see page 1079) to the image list object.

Pascal

```
function Add(Image: TBitmap; Mask: TBitmap): Integer;
```

C++

```
__fastcall int Add(TBitmap Image, TBitmap Mask);
```

Description

Use the Add method to add images with their masks to an image list. It returns the index of the added image, or -1 if it fails. When the Masked (see page 683) property of the image list is false, the second bitmap is ignored; it can be set to nil (Delphi) or NULL (C++).

See Also

AddIcon (see page 655)

AddImages (see page 656)

AddMasked (see page 657)

Masked (see page 683)

19.1.2.5.2 TCustomImageList.AddIcon Method

Adds an icon to the image list object.

Pascal

```
function AddIcon(Image: TIcon): Integer;
```

C++

```
__fastcall int AddIcon(TIcon Image);
```

Description

Use the AddIcon method to add the icon specified by the Image parameter to an image list. AddIcon returns the index of the added image, or -1 if it fails.

See Also

Add (see page 655)

AddImages (see page 656)

AddMasked (see page 657)

19.1.2.5.3 TCustomImageList.AddImage Method

Adds single image from another image list.

Pascal

```
function AddImage(Value: TCustomImageList; Index: Integer): Integer;
```

C++

```
__fastcall int AddImage(TCustomImageList Value, int Index);
```

Description

Use AddImage to add a single image from another image list. Values specifies the source image list. Index indicates the specific image, where images are numbered from 0.

AddImage returns the new size of the image list.

See Also

Add (see page 655)

AddIcon (see page 655)

AddMasked (see page 657)

AddImages (see page 656)

19.1.2.5.4 TCustomImageList.AddImages Method

Adds all images from another image list.

Pascal

```
procedure AddImages(Value: TCustomImageList);
```

C++

```
__fastcall AddImages(TCustomImageList Value);
```

Description

Use AddImages to add all the images from another image list. Value specifies the source image list.

See Also

Add (see page 655)

AddIcon (see page 655)

AddMasked (see page 657)

AddImage (see page 656)

19.1.2.5.5 TCustomImageList.AddMasked Method

Adds an image to the image list along with a color which is used to generate the transparent mask.

Pascal

```
function AddMasked(Image: TBitmap; MaskColor: TColor): Integer;
```

C++

```
__fastcall int AddMasked(TBitmap Image, TColor MaskColor);
```

Description

AddMasked is similar to Add (see page 655) except that a mask bitmap is not specified. Instead, a color that can be used to automatically generate the masks is specified. Each pixel of the specified color in the image bitmap is changed to black, and the corresponding bit in the mask is set to one. As a result, any pixel in the image that matches the specified color is transparent when the image is drawn. AddMasked returns -1 if it fails, 0 if it's successful.

See Also

Add (see page 655)

AddIcon (see page 655)

AddImages (see page 656)

Masked (see page 683)

19.1.2.5.6 TCustomImageList.Assign Method

Copies the information from one image list to another.

Pascal

```
procedure Assign(Source: TPersistent); override;
```

C++

```
virtual __fastcall Assign(TPersistent * Source);
```

Description

Use this method to copy another object's information to the image list.

If Source is another image list (TCustomImageList (see page 651) descendant), Assign discards any current image information and replaces it with the information from Source.

If Source is nil (Delphi) or NULL (C++), Assign clears the image list.

If Source is any other type of object, Assign calls its inherited method, which can copy from any source object that implements a way to copy to an image list in its AssignTo (see page 658) method.

See Also

AddImages (see page 656)

Add (see page 655)

AssignTo

19.1.2.5.7 TCustomImageList.AssignTo Method

Copies the image list's properties to another image list.

Pascal

```
procedure AssignTo(Dest: TPersistent); override;
```

C++

```
virtual __fastcall AssignTo(TPersistent * Dest);
```

Description

Applications do not call the protected AssignTo method. Instead, call the public Assign (see page 657) method of the destination image list. AssignTo is called when the image list is passed as the Source parameter of another object's Assign (see page 657) method and that Assign (see page 657) method can't copy information to the image list. The object whose Assign (see page 657) method was called is passed as the Dest parameter. AssignTo gives the image list an opportunity to copy properties to objects that were not created with the image list in mind.

See Also

AddImages (see page 656)

Add (see page 655)

Assign (see page 657)

19.1.2.5.8 TCustomImageList.Change Method

Generates an OnChange (see page 684) event.

Pascal

```
procedure Change; dynamic;
```

C++

```
__fastcall Change();
```

Description

Change calls the OnChange (see page 684) event handler if it is assigned. Override Change to block this event or provide additional responses to changes in the image list.

See Also

RegisterChanges (see page 672)

OnChange (see page 684)

19.1.2.5.9 TCustomImageList.Clear Method

Clears all images from the image list.

Pascal

```
procedure Clear;
```

C++

```
__fastcall Clear();
```

Description

Use the Clear method to remove all images from the image list. Use the Delete (see page 659) method to delete one image from the image list.

See Also

Delete (see page 659)

19.1.2.5.10 TCustomImageList.DefineProperties Method

Allows bitmap data to be saved and loaded.

Pascal

```
procedure DefineProperties(Filer: TFiler); override;
```

C++

```
virtual __fastcall DefineProperties(TFiler Filer);
```

Description

DefineProperties allows the image list to create pseudo properties that are loaded and saved with the image list. TCustomImageList (see page 651) overrides the inherited method to save or load the bitmap image in addition to the other published properties.

See Also

DefineBinaryProperty

DefineProperties

19.1.2.5.11 TCustomImageList.Delete Method

Deletes a particular image from the image list.

Pascal

```
procedure Delete(Index: Integer);
```

C++

```
__fastcall Delete(int Index);
```

Description

Use the Delete method to delete an image specified by Index in the image list. Use the Clear (see page 658) method to delete all images in the image list.

See Also

Clear (see page 658)

19.1.2.5.12 TCustomImageList.DoDraw Method

Draws the specified image onto the provided Canvas in the indicated style.

Pascal

```
procedure DoDraw(Index: Integer; Canvas: TCanvas; X: Integer; Y: Integer; Style: Cardinal;
Enabled: Boolean = True); virtual;
```

C++

```
virtual __fastcall DoDraw(int Index, TCanvas Canvas, int X, int Y, unsigned Style, Boolean
Enabled = True);
```

Description

Applications can't call the protected DoDraw method. It provides the underlying implementation of the public Draw (see page 660) method.

Index specifies an image to draw.

Canvas indicates the drawing surface on which the image is rendered.

X and Y specify the top-left corner of the image on Canvas.

Style encodes how the image blends color or uses a mask (the DrawingStyle (see page 681) and ImageType (see page 682)).

Enabled indicates whether the image appears as a gray-mapped image. If Enabled is false, DoDraw renders a gray-mapped version of the image.

See Also

Draw (see page 660)

DrawingStyle (see page 681)

ImageType (see page 682)

19.1.2.5.13 Draw Method

This is the overview for the Draw method overload.

Overload List

	Name	Description
	TCustomImageList.Draw (TCanvas, Integer, Integer, Integer, Boolean) (see page 660)	Draws the image specified by the Index parameter onto the provided Canvas.
	TCustomImageList.Draw (TCanvas, Integer, Integer, Integer, TDrawingStyle, TImageType, Boolean) (see page 660)	Draws the image specified by the Index parameter onto the provided Canvas.

19.1.2.5.13.1 TCustomImageList.Draw Method (TCanvas, Integer, Integer, Integer, Boolean)

Draws the image specified by the Index parameter onto the provided Canvas.

Pascal

```
procedure Draw(Canvas: TCanvas; X: Integer; Y: Integer; Index: Integer; Enabled: Boolean =
True); overload;
procedure Draw(Canvas: TCanvas; X: Integer; Y: Integer; Index: Integer; ADrawingStyle:
TDrawingStyle; AImageType: TImageType; Enabled: Boolean = True); overload;
```

C++

```
__fastcall Draw(TCanvas Canvas, int X, int Y, int Index, Boolean Enabled = True);
__fastcall Draw(TCanvas Canvas, int X, int Y, int Index, TDrawingStyle ADrawingStyle,
TImageType AImageType, Boolean Enabled = True);
```

Description

Use the Draw method to draw one of the images in the image list onto a specified canvas.

Canvas is the drawing surface on which to render the image.

X and Y specify the location where the top-left corner should appear on Canvas.

Index indicates which image to draw, where 0 specifies the first image, 1 specifies the second image, and so on.

ADrawingStyle indicates how the color of the image may be altered. If this parameter is not specified, Draw uses the value of the DrawingStyle (see page 681) property.

AImageType indicates whether to draw the image or its associated mask. If this parameter is not specified, Draw uses the value of the ImageType (see page 682) property.

Enabled indicates whether the image appears as a gray-mapped image. If Enabled is false, Draw renders a gray-mapped version of the image. Because Enabled has a default value of true, this parameter can be omitted if the image should not be grayed.

C++ Examples:

```
/*
The following code uses the bitmaps in an image list
component to draw the contents of each cell in a draw grid.
It draws a focus rectangle around the cell that has focus.
goDrawFocusSelect in the DrawGrid Options parameter must be
True to set focus on a cell.  The ImageList Draw method must
be called after DrawFocusRect. The OnSelectCell event handler
must implemented to return true.
*/
void __fastcall TForm1::DrawGrid1DrawCell(
  TObject *Sender, int ACol, int ARow, TRect &Rect, TGridDrawState State)
{
  long index = ARow * DrawGrid1->ColCount + ACol;
  DrawGrid1->Canvas->Brush->Color = clBackground;
  DrawGrid1->Canvas->FillRect(Rect);
  if (State.Contains(gdFocused))
  {
    DrawGrid1->Canvas->DrawFocusRect(Rect);
    Label1->Caption = "Cell " + IntToStr(int(index)) + " has the focus.";
  }
  ImageList1->Draw(
    DrawGrid1->Canvas, Rect.Left, Rect.Top, index, True);
}

void __fastcall TForm1::DrawGrid1SelectCell(TObject *Sender, int ACol, int ARow,
      bool &CanSelect)
{
  CanSelect= True;
}
/*
The following code is the OnDrawPanel event handler of a
status bar.  It draws each panel of the status bar, adding
text and icons stored in ImageList1. The image list contains
as many icons as there are header sections.  This example
requires a populated image list and a statusbar with several
panels added to the Panels property.  Select each panel and
set the Style property to psOwnerDraw.
*/
void __fastcall TForm1::StatusBar1DrawPanel(TStatusBar *StatusBar,
        TStatusPanel *Panel, const TRect &Rect)
{
  TCanvas *canvas = StatusBar->Canvas;
  canvas->Brush->Color = clRed;
  canvas->FillRect(Rect);
  canvas->Font->Color = clYellow;
  ImageList1->Draw(canvas,Rect.Left,Rect.Top, Panel->Index, true);
  canvas->TextOut(Rect.left + 30, Rect.top + 2, "Panel" + IntToStr(Panel->Index));
}
```

Delphi Examples:

```
{
The following code uses the bitmaps in an image list
component to draw the contents of each cell in a draw grid.
It draws a focus rectangle around the cell that has focus.
goDrawFocusSelect in the DrawGrid Options parameter must be
True to set focus on a cell.  The ImageList Draw method must
be called after DrawFocusRect. The OnSelectCell event handler
must implemented to return true.
}
procedure TForm1.DrawGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
var
  index: Integer;
begin
  index := ARow * DrawGrid1.ColCount + ACol;
  DrawGrid1.Canvas.Brush.Color := clWhite;
  DrawGrid1.Canvas.FillRect(Rect);
  if (gdFocused in State) then
  begin
    DrawGrid1.Canvas.DrawFocusRect(Rect);
    Label1.Caption:= 'Cell ' + InttoStr(index) + ' has the focus.';
  end;
  ImageList1.Draw(DrawGrid1.Canvas,Rect.Left,Rect.Top,index, True);
end;

procedure TForm1.DrawGrid1SelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
begin
  CanSelect:= True;
end;
{
The following code is the OnDrawPanel event handler of a
status bar.  It draws each panel of the status bar, adding
text and icons stored in ImageList1. The image list contains
as many icons as there are header sections.  This example
requires a populated image list and a statusbar with several
panels added to the Panels property.  Select each panel and
set the Style property to psOwnerDraw.
}
procedure TForm1.StatusBar1DrawPanel(StatusBar: TStatusBar;
  Panel: TStatusPanel; const Rect: TRect);
begin
with StatusBar1.Canvas do
  begin
    Brush.Color := clRed;
    FillRect(Rect);
    Font.Color := clYellow;
    ImageList1.Draw(StatusBar1.Canvas,Rect.Left,Rect.Top,Panel.Index);
    TextOut(Rect.left + 30, Rect.top + 2, 'Panel' + IntToStr(Panel.Index));
  end;
end;
```

See Also

DrawingStyle (see page 681)

DrawOverlay (see page 663)

GetImages (see page 666)

ImageType (see page 682)

19.1.2.5.14 DrawOverlay Method

This is the overview for the DrawOverlay method overload.

Overload List

	Name	Description
	TCustomImageList.DrawOverlay (TCanvas, Integer, Integer, Integer, TOverlay, Boolean) (see page 663)	Draws an image and an overlay onto the provided Canvas.
	TCustomImageList.DrawOverlay (TCanvas, Integer, Integer, Integer, TOverlay, TDrawingStyle, TImageType, Boolean) (see page 663)	Draws an image and an overlay onto the provided Canvas.

19.1.2.5.14.1 TCustomImageList.DrawOverlay Method (TCanvas, Integer, Integer, Integer, TOverlay, Boolean)

Draws an image and an overlay onto the provided Canvas.

Pascal

```
procedure DrawOverlay(Canvas: TCanvas; X: Integer; Y: Integer; ImageIndex: Integer; Overlay:
TOverlay; Enabled: Boolean = True); overload;
procedure DrawOverlay(Canvas: TCanvas; X: Integer; Y: Integer; ImageIndex: Integer; Overlay:
TOverlay; ADrawingStyle: TDrawingStyle; AImageType: TImageType; Enabled: Boolean = True);
overload;
```

C++

```
__fastcall DrawOverlay(TCanvas Canvas, int X, int Y, int ImageIndex, TOverlay Overlay, Boolean
Enabled = True);
__fastcall DrawOverlay(TCanvas Canvas, int X, int Y, int ImageIndex, TOverlay Overlay,
TDrawingStyle ADrawingStyle, TImageType AImageType, Boolean Enabled = True);
```

Description

An overlay mask is an image drawn transparently over another image. Any image can be used as an overlay mask. To draw an overlay mask over an image use the DrawOverlay method with the index of the overlay mask.

Canvas is the drawing surface on which to draw the overlay.

X and Y specify the location where the top-left corner should appear on Canvas.

ImageIndex indicates which image to draw, where 0 specifies the first image, 1 specifies the second image, and so on.

Overlay (see page 671) specifies the index of the overlay mask to use. This index is assigned to the image that is used as an overlay mask by calling the Overlay (see page 671) method.

ADrawingStyle indicates how the color of the image may be altered. If this parameter is not specified, Draw (see page 660) uses the value of the DrawingStyle (see page 681) property.

AImageType indicates whether to draw the image or its associated mask. If this parameter is not specified, Draw (see page 660) uses the value of the ImageType (see page 682) property.

Enabled indicates whether the image appears as a gray-mapped image. If Enabled is false, DoDraw (see page 659) renders a gray-mapped version of the image. Because Enabled has a default value of true, this parameter can be omitted if the image should not be grayed.

See Also

Overlay (see page 671)

Draw (see page 660)

SetDragImage

19.1.2.5.15 TCustomImageList.FileLoad Method

Loads a resourced image into the image list.

Pascal

```
function FileLoad(ResType: TResType; const Name: string; MaskColor: TColor): Boolean;
```

C++

```
__fastcall Boolean FileLoad(TResType ResType, const UnicodeString Name, TColor MaskColor);
```

Description

Use the FileLoad method to load a bitmap, cursor, or icon from a resource file. Use ResType to specify the type of resource. Use Name to identify the resource by name. Use MaskColor to set the transparent color for the images. FileLoad returns true if successful.

Note: Within a package, FileLoad returns an error. GetInstRes (see page 667) must be used to load a resource from within a package.

See Also

GetResource (see page 668)

ResourceLoad (see page 674)

GetInstRes (see page 667)

19.1.2.5.16 TCustomImageList.GetBitmap Method

Retrieves a specified image as a bitmap.

Pascal

```
function GetBitmap(Index: Integer; Image: TBitmap): Boolean;
```

C++

```
__fastcall Boolean GetBitmap(int Index, TBitmap Image);
```

Description

Use the GetBitmap method to obtain a particular image in the image list as a bitmap object. Index denotes the index of the image. GetBitmap assigns the image to the TBitmap object passed in as the Image parameter.

C++ Examples:

```
/*
This example assigns a bitmap contained in an ImageList to
an Image component's Picture's Bitmap property and displays
the bitmap.  TImage Stretch will fit the bitmap to the image
size.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  ImageList1->GetBitmap(0, Image1->Picture->Bitmap);
  Image1->Stretch = True;
```

```
}
```

Delphi Examples:

```
{
This example assigns a bitmap contained in an ImageList to
an Image component's Picture's Bitmap property and displays
the bitmap.  TImage Stretch will fit the bitmap to the image
size.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
   ImageList1.GetBitmap(2,Image1.Picture.Bitmap);
   Image1.Stretch := True;
end;
```

See Also

GetMaskBitmap (see page 668)

19.1.2.5.17 TCustomImageList.GetHotSpot Method

Returns the hot spot location associated with the current drag image.

Pascal

```
function GetHotSpot: TPoint; virtual;
```

C++

```
virtual __fastcall TPoint GetHotSpot();
```

Description

Use the GetHotSpot method to obtain the position of the hot spot of the current drag image.

See Also

DragMove

SetDragImage

19.1.2.5.18 GetIcon Method

This is the overview for the GetIcon method overload.

Overload List

	Name	Description
	TCustomImageList.GetIcon (Integer, TIcon) (see page 665)	Returns the image specified by Index as an icon into the Image parameter.
	TCustomImageList.GetIcon (Integer, TIcon, TDrawingStyle, TImageType) (see page 665)	Returns the image specified by Index as an icon into the Image parameter.

19.1.2.5.18.1 TCustomImageList.GetIcon Method (Integer, TIcon)

Returns the image specified by Index as an icon into the Image parameter.

Pascal

```
procedure GetIcon(Index: Integer; Image: TIcon); overload;
procedure GetIcon(Index: Integer; Image: TIcon; ADrawingStyle: TDrawingStyle; AImageType:
TImageType); overload;
```

C++

```
__fastcall GetIcon(int Index, TIcon Image);
__fastcall GetIcon(int Index, TIcon Image, TDrawingStyle ADrawingStyle, TImageType AImageType);
```

Description

Call GetIcon to obtain a particular image in the image list as an icon.

Index identifies the desired image, where 0 specifies the first image, 1 specifies the second image, and so on.

Image is a TIcon object that receives the new image.

ADrawingStyle specifies the amount of color that is blended with the image to achieve various affects. When GetIcon is called using the version that does not take an ADrawingStyle parameter, it uses the value of the DrawingStyle (see page 681) property.

AImageType specifies whether the image is rendered using a mask. When GetIcon is called using the version that does not take an AImageType parameter, it uses the value of the ImageType (see page 682) property.

See Also

GetImageBitmap (see page 666)

DrawingStyle (see page 681)

ImageType (see page 682)

19.1.2.5.19 TCustomImageList.GetImageBitmap Method

Returns a handle to a bitmap containing all of the images in the image list.

Pascal

```
function GetImageBitmap: HBITMAP;
```

C++

```
__fastcall HBITMAP GetImageBitmap();
```

Description

Use the GetImageBitmap method to obtain a Windows handle for the bitmap that contains all images in the image list. GetImageBitmap returns 0 if unsuccessful.

See Also

GetImages (see page 666)

GetBitmap (see page 664)

19.1.2.5.20 TCustomImageList.GetImages Method

Returns a specified image and its mask as bitmaps.

Pascal

```
procedure GetImages(Index: Integer; Image: TBitmap; Mask: TBitmap);
```

C++

```
__fastcall GetImages(int Index, TBitmap Image, TBitmap Mask);
```

Description

Use GetImages to obtain the image (and its mask) identified by the Index parameter.

Index specifies the image (and mask) to return, where 0 identifies the first image, 1 the second image, and so on.

Image is a TBitmap object that receives the image of the bitmap.

Mask (see page 1079) is a TBitmap object that receives the image of the mask.

See Also

GetImageBitmap (see page 666)

GetMaskBitmap (see page 668)

GetBitmap (see page 664)

19.1.2.5.21 GetInstRes Method

This is the overview for the GetInstRes method overload.

Overload List

	Name	Description
	TCustomImageList.GetInstRes (THandle, TResType, DWORD, Integer, TLoadResources, TColor) (see page 667)	Loads a specified bitmap, cursor, or icon resource into the image list.
	TCustomImageList.GetInstRes (THandle, TResType, string, Integer, TLoadResources, TColor) (see page 667)	Loads a specified bitmap, cursor, or icon resource into the image list.

19.1.2.5.21.1 TCustomImageList.GetInstRes Method (THandle, TResType, DWORD, Integer, TLoadResources, TColor)

Loads a specified bitmap, cursor, or icon resource into the image list.

Pascal

```
function GetInstRes(Instance: THandle; ResType: TResType; const Name: string; Width: Integer;
LoadFlags: TLoadResources; MaskColor: TColor): Boolean; overload;
function GetInstRes(Instance: THandle; ResType: TResType; ResID: DWORD; Width: Integer;
LoadFlags: TLoadResources; MaskColor: TColor): Boolean; overload;
```

C++

```
__fastcall Boolean GetInstRes(THandle Instance, TResType ResType, const UnicodeString Name,
int Width, TLoadResources LoadFlags, TColor MaskColor);
__fastcall Boolean GetInstRes(THandle Instance, TResType ResType, DWORD ResID, int Width,
TLoadResources LoadFlags, TColor MaskColor);
```

Description

Use the GetInstRes method to load bitmap, cursor, or icon resource into an image list.

Instance is the instance handle of the resource module.

ResType indicates the type of resource (rtBitmap, rtCursor or rtIcon).

Name or ResID identifies the resource by name or ID, respectively.

Width (see page 684) specifies the width, in pixels, of the image.

LoadFlags specifies how to load the image.

MaskColor sets the transparent color for the image.

Note: GetInstRes must be used to load a resource from within a package. Within a package, GetResource (see page 668) or FileLoad (see page 664) returns an error.

See Also

ResInstLoad (see page 673)

ResourceLoad (see page 674)

FileLoad (see page 664)

GetResource (see page 668)

19.1.2.5.22 TCustomImageList.GetMaskBitmap Method

Returns a handle to a bitmap containing all masks for images in the image list.

Pascal

```
function GetMaskBitmap: HBITMAP;
```

C++

```
__fastcall HBITMAP GetMaskBitmap();
```

Description

Use the GetMaskBitmap to obtain a bitmap that contains all masks in the image list. GetMaskBitmap returns 0 if unsuccessful.

See Also

GetImageBitmap (see page 666)

GetImages (see page 666)

19.1.2.5.23 TCustomImageList.GetResource Method

Loads the specified bitmap, cursor, or icon resource into the image list.

Pascal

```
function GetResource(ResType: TResType; const Name: string; Width: Integer; LoadFlags:
TLoadResources; MaskColor: TColor): Boolean;
```

C++

```
__fastcall Boolean GetResource(TResType ResType, const UnicodeString Name, int Width,
TLoadResources LoadFlags, TColor MaskColor);
```

Description

Use the GetResource method to load bitmap, cursor, or icon resource into an image list.

Note: Within a package, GetResource returns an error. GetInstRes (see page 667) must be used to load a resource from within a package.

ResType specifies the type of graphical resource that is to be loaded into the image list and can be rtBitmap, rtCursor or rtIcon.

Name identifies the resource by name.

Width (see page 684) specifies the width, in pixels, of the image.

LoadFlags specifies how to load the image.

MaskColor sets the transparent color for the images in the file.

See Also

ResourceLoad (see page 674)

FileLoad (see page 664)

GetInstRes (see page 667)

19.1.2.5.24 TCustomImageList.HandleAllocated Method

Indicates whether a window handle exists for the image list.

Pascal

```
function HandleAllocated: Boolean;
```

C++

```
__fastcall Boolean HandleAllocated();
```

Description

Use HandleAllocated to determine if a handle exists for the image list. HandleAllocated returns true if the Windows image list has been created and a handle is available, false otherwise. HandleAllocated doesn't create a handle if it doesn't already exist.

Note: Getting the Handle (see page 682) property of creates the Windows image list automatically if it didn't previously exist, and returns the resulting handle.

See Also

Handle (see page 682)

19.1.2.5.25 TCustomImageList.HandleNeeded Method

Creates an image list handle for the image list if one doesn't already exist.

Pascal

```
procedure HandleNeeded;
```

C++

```
__fastcall HandleNeeded();
```

Description

Use HandleNeeded to create a Windows image list for the image list component when one doesn't already exist. This method is called automatically when an image list's handle property is accessed.

See Also

HandleAllocated (see page 669)

Handle (see page 682)

19.1.2.5.26 TCustomImageList.Initialize Method

Initializes a newly allocated image list.

Pascal

```
procedure Initialize; virtual;
```

C++

```
virtual __fastcall Initialize();
```

Description

Initialize is called from the constructor of the image list. It sets numerous image list properties to their default values and creates instances of helper objects used internally.

See Also

Create (see page 653)

CreateSize (see page 653)

19.1.2.5.27 TCustomImageList.Insert Method

Inserts a bitmap and its mask into the image list following a specified image.

Pascal

```
procedure Insert(Index: Integer; Image: TBitmap; Mask: TBitmap);
```

C++

```
__fastcall Insert(int Index, TBitmap Image, TBitmap Mask);
```

Description

Use the Insert method to insert a bitmap and its mask into the image list. The image is inserted following the image specified by Index.

See Also

Add (see page 655)

AddMasked (see page 657)

19.1.2.5.28 TCustomImageList.InsertIcon Method

Inserts an icon at a specified position in the image list.

Pascal

```
procedure InsertIcon(Index: Integer; Image: TIcon);
```

C++

```
__fastcall InsertIcon(int Index, TIcon Image);
```

Description

Use the InsertIcon method to insert an icon into the image list. The icon is inserted following the image specified by Index.

See Also

AddIcon (see page 655)

19.1.2.5.29 TCustomImageList.InsertMasked Method

Inserts a bitmap at a specified position, creating the mask from a specified transparent color.

Pascal

```
procedure InsertMasked(Index: Integer; Image: TBitmap; MaskColor: TColor);
```

C++

```
__fastcall InsertMasked(int Index, TBitmap Image, TColor MaskColor);
```

Description

Use the InsertMasked method to insert a masked image into the image list. The new image is inserted following the image specified by Index. Use MaskColor to identify the color in the image used to generate the Mask (see page 1079).

See Also

AddMasked (see page 657)

Insert (see page 670)

19.1.2.5.30 TCustomImageList.Move Method

Moves a specified image to another position.

Pascal

```
procedure Move(CurIndex: Integer; NewIndex: Integer);
```

C++

```
__fastcall Move(int CurIndex, int NewIndex);
```

Description

Use the Move method to move an image to another position in the image list. Specify the current position using CurIndex, and the new position using NewIndex.

See Also

Insert (see page 670)

19.1.2.5.31 TCustomImageList.Overlay Method

Identifies an image in the list as an image that can be used as an overlay mask.

Pascal

```
function Overlay(ImageIndex: Integer; Overlay: TOverlay): Boolean;
```

C++

```
__fastcall Boolean Overlay(int ImageIndex, TOverlay Overlay);
```

Description

Use Overlay to assign an overlay index to one of the images in the list. Once an image has an overlay index, it can be used as an overlay mask. An overlay mask is an image drawn transparently over another image.

ImageIndex identifies the image that can act as an overlay mask.

Overlay is the overlay index to assign to that image. It must be a value between 0 and 3 (inclusive). Note that this means at most four images can be used as overlay masks.

If successful, Overlay returns true.

To draw an overlay mask over an image use the DrawOverlaymethodwith the index of the overlay mask.

See Also

DrawOverlay (see page 663)

19.1.2.5.32 TCustomImageList.ReadData Method

Pascal

```
procedure ReadData(Stream: TStream); virtual;
```

C++

```
virtual __fastcall ReadData(TStream Stream);
```

Description

This is ReadData, a member of class TCustomImageList.

19.1.2.5.33 TCustomImageList.RegisterChanges Method

Registers an object to be notified when changes occur in the image list.

Pascal

```
procedure RegisterChanges(Value: TChangeLink);
```

C++

```
__fastcall RegisterChanges(TChangeLink Value);
```

Description

Use the RegisterChanges method to have an object notified whenever changes occur in the image list. The TChangeLink (see page 648)'s OnChange (see page 684) event is called whenever a change in the image list occurs.

Create (see page 653) a change link object for each object that should be notified of changes. Register each object by calling RegisterChanges and passing its change link object as the Value parameter. The OnChange (see page 684) event of all registered change link objects is then triggered when a change occurs in the image list. Any code assigned to the OnChange (see page 684) event handlers will execute.

See Also

UnRegisterChanges (see page 674)

OnChange (see page 650)

19.1.2.5.34 TCustomImageList.Replace Method

Replaces an image in the image list with a new image and mask.

Pascal

```
procedure Replace(Index: Integer; Image: TBitmap; Mask: TBitmap);
```

C++

```
__fastcall Replace(int Index, TBitmap Image, TBitmap Mask);
```

Description

Use the Replace method to replace the image at the Index position in the image list with a new Image and overlay Mask (see page 1079).

See Also

Insert (see page 670)

ReplaceIcon (see page 673)

ReplaceMasked (see page 673)

19.1.2.5.35 TCustomImageList.ReplaceIcon Method

Replaces an image in the image list with an icon.

Pascal

```
procedure ReplaceIcon(Index: Integer; Image: TIcon);
```

C++

```
__fastcall ReplaceIcon(int Index, TIcon Image);
```

Description

Use the ReplaceIcon method to replace the Image at the Index position in the image list with a new icon.

See Also

Replace (see page 672)

InsertIcon (see page 670)

19.1.2.5.36 TCustomImageList.ReplaceMasked Method

Replaces an image in the image list with a new image and automatically-generated mask.

Pascal

```
procedure ReplaceMasked(Index: Integer; NewImage: TBitmap; MaskColor: TColor);
```

C++

```
__fastcall ReplaceMasked(int Index, TBitmap NewImage, TColor MaskColor);
```

Description

Use the ReplaceMasked method to replace the image at Index position within the image list specified by NewImage and the mask generated from NewImage by masking out all pixels of the color specified by MaskColor.

See Also

Replace (see page 672)

InsertMasked (see page 670)

19.1.2.5.37 TCustomImageList.ResInstLoad Method

Loads a resource into the image list.

Pascal

```
function ResInstLoad(Instance: THandle; ResType: TResType; const Name: string; MaskColor:
TColor): Boolean;
```

C++

```
__fastcall Boolean ResInstLoad(THandle Instance, TResType ResType, const UnicodeString Name,
TColor MaskColor);
```

Description

Use the ResoInstLoad method to load a resource of type TResType (see page 688) into the image list. Use Instance to specify the instance handle of the resource module, and Name to identify the resource within that module. MaskColor is used to create the mask for the resulting image. ResInstLoad should be used rather than ResourceLoad (see page 674) to load a resource

from within a package.

ResInstLoad returns true if it is successful, false otherwise.

See Also

GetInstRes (see page 667)

GetResource (see page 668)

19.1.2.5.38 TCustomImageList.ResourceLoad Method

Loads a resource into the image list.

Pascal

```
function ResourceLoad(ResType: TResType; const Name: string; MaskColor: TColor): Boolean;
```

C++

```
__fastcall Boolean ResourceLoad(TResType ResType, const UnicodeString Name, TColor MaskColor);
```

Description

Use the ResourceLoad method to load a resource of type TResType (see page 688) into the image list. Use Name to identify the resource. MaskColor is used to create the mask for the resulting image. ResourceLoad returns true if successful, false otherwise.

Note: Within a package, use ResInstLoad (see page 673) instead of ResourceLoad. ResourceLoad returns an error when called from within a package.

See Also

GetResource (see page 668)

ResInstLoad (see page 673)

19.1.2.5.39 TCustomImageList.SetSize Method

Pascal

```
procedure SetSize(AWidth: Integer; AHeight: Integer);
```

C++

```
__fastcall SetSize(int AWidth, int AHeight);
```

Description

This is SetSize, a member of class TCustomImageList.

19.1.2.5.40 TCustomImageList.UnRegisterChanges Method

Removes an object from the list of objects to be notified when the image list changes.

Pascal

```
procedure UnRegisterChanges(Value: TChangeLink);
```

C++

```
__fastcall UnRegisterChanges(TChangeLink Value);
```

Description

Use the UnRegisterChanges method to unregister a TChangeLink (see page 648) object. The object will no longer receive

notification when the image list changes.

See Also

RegisterChanges (see page 672)

19.1.2.5.41 TCustomImageList.WriteData Method

Pascal

```
procedure WriteData(Stream: TStream); virtual;
```

C++

```
virtual __fastcall WriteData(TStream Stream);
```

Description

This is WriteData, a member of class TCustomImageList.

19.1.2.6 TCustomImageList Properties

The properties of the TCustomImageList class are listed here.

Public Properties

	Name	Description
	AllocBy (see page 675)	Sets the number of images by which the image list grows when it needs to make room for new images.
	BkColor (see page 676)	Determines which background color to use when drawing an image.
	BlendColor (see page 676)	Determines which foreground color to use when drawing an image.
R	Count (see page 677)	Indicates the number of images in the image list.
	DrawingStyle (see page 681)	Specifies the style to be used when the image list is drawing an image.
	Handle (see page 682)	Designates the handle for the image list.
	Height (see page 682)	Specifies the height of images within the image list.
	ImageType (see page 682)	Determines whether the image list uses the associated mask when drawing.
	Masked (see page 683)	Specifies whether the image list includes masks that can be combined with the images.
	ShareImages (see page 683)	Specifies whether the image list destroys its handle when the image list is destroyed.
	Width (see page 684)	Specifies the width of all images within the image list.

Published Properties

	Name	Description
	ColorDepth (see page 677)	Color depth of images in list.

19.1.2.6.1 TCustomImageList.AllocBy Property

Sets the number of images by which the image list grows when it needs to make room for new images.

Pascal

```
property AllocBy: Integer;
```

C++

```
__property int AllocBy;
```

Description

Use the AllocBy method to specify the number of new images that can be added to the resized image list.

See Also

Count (see page 677)

Add (see page 655)

AddIcon (see page 655)

AddImages (see page 656)

AddMasked (see page 657)

19.1.2.6.2 TCustomImageList.BkColor Property

Determines which background color to use when drawing an image.

Pascal

```
property BkColor: TColor;
```

C++

```
__property TColor BkColor;
```

Description

Use BkColor to specify the color used for the masked regions of the images in the list. When BkColor is clNone, the image is drawn transparently using the mask to define the transparent areas. Otherwise, the value of BkColor appears in the masked out portions.

Note: BkColor has no effect when the Masked (see page 683) property is false.

See Also

BlendColor (see page 676)

Masked (see page 683)

ImageType (see page 682)

19.1.2.6.3 TCustomImageList.BlendColor Property

Determines which foreground color to use when drawing an image.

Pascal

```
property BlendColor: TColor;
```

C++

```
__property TColor BlendColor;
```

Description

Set BlendColor to cause the images in the list to be combined with the specified color when the DrawingStyle (see page 681) is dsFocus or dsSelected. More of BlendColor is combined with the images when DrawingStyle (see page 681) is dsSelected. clNone specifies no blend color while clDefault specifies the system highlight .

See Also

BkColor (see page 676)

DrawingStyle (see page 681)

19.1.2.6.4 TCustomImageList.ColorDepth Property

Color depth of images in list.

Pascal

```
property ColorDepth: TColorDepth;
```

C++

```
__property TColorDepth ColorDepth;
```

Description

ColorDepth is the color depth type TColorDepth (see page 685) of images in the list.

See Also

BkColor (see page 676)

BlendColor (see page 676)

Height (see page 682)

Width (see page 684)

TColorDepth (see page 685)

19.1.2.6.5 TCustomImageList.Count Property

Indicates the number of images in the image list.

Pascal

```
property Count: Integer;
```

C++

```
__property int Count;
```

Description

Use Count when iterating through the images in the image list. Retrieve the individual images using the GetBitmap (see page 664) method. Draw (see page 660) individual images using the Draw (see page 660) method.

C++ Examples:

```
/*
This example requires a TListView, a TImageList and a
TComboBox. You will need to double click the image list and
add several images to the image list prior to running the
project. During the form's OnCreate event
handler, items for the List View control are created for
each image in the Image List and the ImageIndex is assigned
the number of the image within the ImageList. Two columns are
created so that when ViewStyle is vsReport, you will have
columns to view.  Also within the form's OnCreate event
handler assign the ComboBox each of the four TViewStyle
constants to the Items' Objects property. You could also
simply code this within a series of OnClick event handlers
as, for instance, ListView1.->ViewStyle := vsIcon.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  TListItem *Item;
  TListColumn *Column;
  const char imagenames[3][20] = {"C++ Image", "Borland Image", "Delphi Image"};
```

```
  const char Col2Array[3][40] = {"Documentation for the C++ icon.", "Borland icon.", "Delphi
icon."};
  // Create a ListView item for each image in the ImageList
  ListView1->SmallImages = ImageList1;
  ListView1->LargeImages = ImageList1;
  for (int i = 0; i < ImageList1->Count; i++)
  {
    Item = ListView1->Items->Add();
    Item->Caption = imagenames[i];
    Item->ImageIndex = i;
    Item->SubItems->Add(Col2Array[i]);
  }
  // Create two columns to show during viewing as vsReport
  Column = ListView1->Columns->Add();
  Column->Caption = "Column 1";
  Column->Width = 200;
  Column = ListView1->Columns->Add();
  Column->Caption = "Column 2";
  Column->Width = 200;
  // Add View styles and constants to the Combo Box
  ComboBox1->Items->AddObject("vsIcon", reinterpret_cast<TObject *>(vsIcon));
  ComboBox1->Items->AddObject("vsList", reinterpret_cast<TObject *>(vsList));
  ComboBox1->Items->AddObject("vsReport", reinterpret_cast<TObject *>(vsReport));
  ComboBox1->Items->AddObject("vsSmallIcon", reinterpret_cast<TObject *>(vsSmallIcon));
  // Display first item in the Combo Box
  ComboBox1->ItemIndex = 0;
}

void __fastcall TForm1::ComboBox1Click(TObject *Sender)
{
  ListView1->ViewStyle = (TViewStyle) ComboBox1->Items->Objects[ComboBox1->ItemIndex];
}
/*
This example shows how to use the OnColumnClick and
OnCompare events of a list view to let users sort the
columns in a report-style list view by clicking on the
column headers. This requires a global variable to keep
track of the column that was clicked.  The OnColumnClick
event handler sets the global variable to indicate the
column to sort and calls AlphaSort:
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  TListItem *Item;
  TListColumn *Column;
  const char imagenames[3][20] = {
    "C++ Image", "Borland Image", "Delphi Image"};
  const char Col2Array[3][40] = {
    "Documentation for the C++ icon.",
    "Borland icon.", "Delphi icon."
    };
  // Create a ListView item for each image in the ImageList
  ListView1->SmallImages = ImageList1;
  ListView1->LargeImages = ImageList1;
  for (int i = 0; i < ImageList1->Count; i++)
  {
    Item = ListView1->Items->Add();
    Item->Caption = imagenames[i];
    Item->ImageIndex = i;
    Item->SubItems->Add(Col2Array[i]);
  }
  // Create two columns to show during viewing as vsReport
  Column = ListView1->Columns->Add();
  Column->Caption = "Column 1";
  Column->Width = 200;
  Column = ListView1->Columns->Add();
```

```
  Column->Caption = "Column 2";
  Column->Width = 200;
  ListView1->ViewStyle = vsReport;
}

int ColumnToSort = 0;

/*
The OnColumnClick event handler sets the global variable to
indicate the column to sort and calls AlphaSort:
*/

void __fastcall TForm1::ListView1ColumnClick(TObject *Sender,
      TListColumn *Column)
{
  ColumnToSort = Column->Index;
  dynamic_cast<TCustomListView *>(Sender)->AlphaSort();
}

/*
The OnCompare event handler causes the list view to sort on
the selected column:
*/

void __fastcall TForm1::ListView1Compare(TObject *Sender, TListItem *Item1,
      TListItem *Item2, int Data, int &Compare)
{
  if (ColumnToSort == 0)
    Compare = CompareText(Item1->Caption,Item2->Caption);
  else
  {
    int ix = ColumnToSort - 1;
    Compare =
      CompareText(Item1->SubItems->Strings[ix], Item2->SubItems->Strings[ix]);
  }
}
```

Delphi Examples:

```
{
This example requires a TListView, a TImageList and a
TComboBox. You will need to double click the image list and
add several images to the image list prior to running the
project. You can use *.bmp or *.ico files from the
\Images\Icons directory. During the form's OnCreate event
handler, items for the List View control are created for
each image in the Image List and the ImageIndex is assigned
the number of the image within the ImageList. Two columns are
created so that when ViewStyle is vsReport, you will have
columns to view.  Also within the form's OnCreate event
handler assign the ComboBox each of the four TViewStyle
constants to the Items' Objects property. You could also
simply code this within a series of OnClick event handlers
as, for instance, ListView1.->ViewStyle := vsIcon.
}

procedure TComboForm.FormCreate(Sender: TObject);
var
  I: Integer;
  ListItem: TListItem;
  NewColumn: TListColumn;
begin
  Col2Array[0]:= 'Documentation for Image0.';
  Col2Array[1]:= 'Documentation for Image1.';
  Col2Array[2]:= 'Documentation for Image2.';
  // Create a ListView item for each image in the ImageList
  with ListView1 do
```

```
    begin
      SmallImages := ImageList1;
      LargeImages := ImageList1;
      for I := 0 to ImageList1.Count - 1 do
      begin
        ListItem := Items.Add;
        Listitem.Caption := 'Image' + IntToStr(I);
        ListItem.ImageIndex := I;
        Listitem.SubItems.Add(Col2Array[I]);
      end;
      // Create two columns to show during viewing as vsReport
      NewColumn := Columns.Add;
      NewColumn.Caption := 'Column 1';
      NewColumn.Width := 200;
      NewColumn := Columns.Add;
      NewColumn.Caption := 'Column 2';
      NewColumn.Width := 200;
      // Add View styles and constants to the Combo Box
      ComboBox1.Items.AddObject('vsIcon', TObject(vsIcon));
      ComboBox1.Items.AddObject('vsList', TObject(vsList));
      ComboBox1.Items.AddObject('vsReport', TObject(vsReport));
      ComboBox1.Items.AddObject(
        'vsSmallIcon', TObject(vsSmallIcon));
      // Display first item in the Combo Box
      ComboBox1.ItemIndex := 0;
    end;
  end;

  procedure TComboForm.ComboBox1Click(Sender: TObject);
  begin
    with ComboBox1 do
      ListView1.ViewStyle := TViewStyle(Items.Objects[ItemIndex]);

  end;
  {
  This example shows how to use the OnColumnClick and
  OnCompare events of a list view to let users sort the
  columns in a report-style list view by clicking on the
  column headers. This requires a global variable to keep
  track of the column that was clicked.  The OnColumnClick
  event handler sets the global variable to indicate the
  column to sort and calls AlphaSort:
  }
  var ColumnToSort: Integer;

  procedure TForm1.FormCreate(Sender: TObject);
  var
    I: Integer;
    ListItem: TListItem;
    NewColumn: TListColumn;
  const
    imagenames: Array[0..2] of string = ('C++ Image', 'Borland Image', 'Delphi Image');
  begin
    Col2Array[0]:= 'Documentation for the C++ icon.';
    Col2Array[1]:= 'Borland icon.';
    Col2Array[2]:= 'Delphi icon.';
    // Create a ListView item for each image in the ImageList
    with ListView1 do
    begin
      SmallImages := ImageList1;
      LargeImages := ImageList1;
      for I := 0 to ImageList1.Count - 1 do
      begin
        ListItem := Items.Add;
        Listitem.Caption := imagenames[I];
        ListItem.ImageIndex := I;
```

```
        Listitem.SubItems.Add(Col2Array[I]);
      end;
      // Create two columns to show during viewing as vsReport
      NewColumn := Columns.Add;
      NewColumn.Caption := 'Column 1';
      NewColumn.Width := 200;
      NewColumn := Columns.Add;
      NewColumn.Caption := 'Column 2';
      NewColumn.Width := 200;
      ListView1.ViewStyle := vsReport;
    end;
  end;

  procedure TForm1.ListView1ColumnClick(Sender: TObject; Column: TListColumn);
  begin
    ColumnToSort := Column.Index;
    (Sender as TCustomListView).AlphaSort;
  end;

  {
  The OnCompare event handler causes the list view to sort on
  the selected column:
  }
  procedure TForm1.ListView1Compare(Sender: TObject; Item1, Item2: TListItem; Data: Integer; var
  Compare: Integer);
  var
    ix: Integer;
  begin
    if ColumnToSort = 0 then
      Compare := CompareText(Item1.Caption,Item2.Caption)
    else begin
     ix := ColumnToSort - 1;
     Compare := CompareText(Item1.SubItems[ix],Item2.SubItems[ix]);
    end;
  end;
  {
  Note:   This OnCompare event handler uses the global
  CompareText function. An application may want to use
  AnsiCompareText, CompareStr, or AnsiCompareStr instead,
  depending on whether the comparison should be case-sensitive
  and whether the locale should be considered.
  }
```

See Also

AllocBy (see page 675)

GetBitmap (see page 664)

Draw (see page 660)

19.1.2.6.6 TCustomImageList.DrawingStyle Property

Specifies the style to be used when the image list is drawing an image.

Pascal

```
property DrawingStyle: TDrawingStyle;
```

C++

```
__property TDrawingStyle DrawingStyle;
```

Description

Use the DrawingStyle property to control how the image is to appear. DrawingStyle controls how much of the blend color specified by BlendColor (see page 676) is combined with the foreground, or what background color to use.

See Also

Draw (see page 660)

DrawOverlay (see page 663)

ImageType (see page 682)

BlendColor (see page 676)

BkColor (see page 676)

19.1.2.6.7 TCustomImageList.Handle Property

Designates the handle for the image list.

Pascal

```
property Handle: HImageList;
```

C++

```
__property HImageList Handle;
```

Description

Handle is the Windows image list object handle. Use this handle when calling Windows API functions that require it.

See Also

HandleAllocated (see page 669)

19.1.2.6.8 TCustomImageList.Height Property

Specifies the height of images within the image list.

Pascal

```
property Height: Integer;
```

C++

```
__property int Height;
```

Description

Use the Height property to specify the height of images before any are added to the image list. The image list is cleared when this value changes.

See Also

Width (see page 684)

CreateSize (see page 653)

Create (see page 653)

19.1.2.6.9 TCustomImageList.ImageType Property

Determines whether the image list uses the associated mask when drawing.

Pascal

```
property ImageType: TImageType;
```

C++

```
__property TImageType ImageType;
```

Description

Use the ImageType property to control whether the images are drawn using the image mask (itMask), or without the mask (itImage).

See Also

DrawingStyle (see page 681)

Draw (see page 660)

Masked (see page 683)

DrawOverlay (see page 663)

19.1.2.6.10 TCustomImageList.Masked Property

Specifies whether the image list includes masks that can be combined with the images.

Pascal

```
property Masked: Boolean;
```

C++

```
__property Boolean Masked;
```

Description

Set the Masked property to true to combine the images with masks. The masked portion of the bitmap is drawn transparently, or replaced by the color specified by the BkColor (see page 676) property.

Masked indicates whether the image list stores masks to combine with the images. To use those masks, the ImageType (see page 682) property indicates whether to draw the image or the mask.

See Also

AddMasked (see page 657)

InsertMasked (see page 670)

BkColor (see page 676)

ImageType (see page 682)

19.1.2.6.11 TCustomImageList.ShareImages Property

Specifies whether the image list destroys its handle when the image list is destroyed.

Pascal

```
property ShareImages: Boolean;
```

C++

```
__property Boolean ShareImages;
```

Description

Set the ShareImages property to false to ensure that the handle to the image list is freed when the image list is destroyed.

See Also

Destroy (see page 654)

Handle (see page 682)

19.1.2.6.12 TCustomImageList.Width Property

Specifies the width of all images within the image list.

Pascal

```
property Width: Integer;
```

C++

```
__property int Width;
```

Description

Use the Width property to specify the width of images before any are added to the image list. The image list is cleared when this value changes.

See Also

Height (see page 682)

Create (see page 653)

CreateSize (see page 653)

19.1.2.7 TCustomImageList Events

The events of the TCustomImageList class are listed here.

Public Events

	Name	Description
	OnChange (see page 684)	Occurs when the image list changes.

19.1.2.7.1 TCustomImageList.OnChange Event

Occurs when the image list changes.

Pascal

```
property OnChange: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnChange;
```

Description

Write an OnChange event handler to execute code when the image list changes. Any objects registered with the RegisterChanges (see page 672) method will also be notified.

See Also

RegisterChanges (see page 672)

19.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TColorDepth (see page 685)	Color depth of list images.
	TDrawingStyle (see page 686)	TDrawingStyle indicates the style to use when rendering an image from an image list.
	TImageType (see page 687)	TImageType indicates whether to use a mask when rendering an image from an image list.
	TLoadResource (see page 687)	TLoadResource and TLoadResources (see page 689) determine how a resource is loaded into an image list.
	TResType (see page 688)	TResType identifies a type of graphical resource to an image list.

19.2.1 ImgList.TColorDepth Enumeration

Color depth of list images.

Pascal

```
TColorDepth = (
  cdDefault,
  cdDeviceDependent,
  cd4Bit,
  cd8Bit,
  cd16Bit,
  cd24Bit,
  cd32Bit
);
```

C++

```
enum TColorDepth {
  cdDefault,
  cdDeviceDependent,
  cd4Bit,
  cd8Bit,
  cd16Bit,
  cd24Bit,
  cd32Bit
};
```

File

ImgList

Description

TColorDepth is the color depth of images in the list. This is the number of bits to represent color for each pixel. The following table describes the possible values:

Value	Meaning
cdDefault	Default color depth.
cdDeviceDependent	Device dependent color depth.

cd4Bit	4 bit color depth.
cd8Bit	8 bit color depth.
cd16Bit	16 bit color depth.
cd24Bit	24 bit color depth.
cd32Bit	32 bit color depth.

See Also

ColorDepth (see page 677)

19.2.2 ImgList.TDrawingStyle Enumeration

TDrawingStyle indicates the style to use when rendering an image from an image list.

Pascal

```
TDrawingStyle = (
  dsFocus,
  dsSelected,
  dsNormal,
  dsTransparent
);
```

C++

```
enum TDrawingStyle {
  dsFocus,
  dsSelected,
  dsNormal,
  dsTransparent
};
```

File

ImgList

Description

TDrawingStyle describes how to render an image from an image list component. The following table describes the possible values:

Value	Meaning
dsFocus	Draws the image blending 25% with the color specified by the BlendColor property. This only affects image lists that contain masks.
dsSelected	Draws the image blending 50% with the color specified by the BlendColor property. This only affects image lists that contain masks.
dsNormal	Draws the image using the color specified in the BkColor property. If the BkColor is clNone, the image is drawn transparently using the mask.
dsTransparent	Draws using the mask regardless of the BkColor setting.

19.2.3 ImgList.TImageType Enumeration

TImageType indicates whether to use a mask when rendering an image from an image list.

Pascal

```
TImageType = (
  itImage,
  itMask
);
```

C++

```
enum TImageType {
  itImage,
  itMask
};
```

File

ImgList

Description

TImageType indicates whether to use a mask when rendering an image in an image list. It includes the following values:

Value	Meaning
itImage	Image does not include a mask.
itMask	Image uses a mask.

19.2.4 ImgList.TLoadResource Enumeration

TLoadResource and TLoadResources (see page 689) determine how a resource is loaded into an image list.

Pascal

```
TLoadResource = (
  lrDefaultColor,
  lrDefaultSize,
  lrFromFile,
  lrMap3DColors,
  lrTransparent,
  lrMonoChrome
);
```

C++

```
enum TLoadResource {
  lrDefaultColor,
  lrDefaultSize,
  lrFromFile,
  lrMap3DColors,
  lrTransparent,
  lrMonoChrome
};
```

File

ImgList

Description

TLoadResources (see page 689) is a set of flags (TLoadResource values) that govern the way an image list loads image resources. The following table lists the possible TLoadResource values.

Value	Meaning
lrDefaultColor	Use the color format of the display.
lrDefaultSize	Use system metrics for Height and Width if the Width parameter to GetResource is 0. If the Width parameter is 0 and this option is not specified for cursors and icons, the resources width and height will be used.
lrFromFile	Loads the resource from a specified file.
lrMap3DColors	Replaces the images' gray shades with the 3D windows colors.
lrTransparent	Replaces the color table entry for the first pixel of the resource with the default window color (COLOR_WINDOW). This applies only to images with color tables.
lrMonoChrome	Loads the resource as black and white.

See Also

SelectDirectory

19.2.5 ImgList.TResType Enumeration

TResType identifies a type of graphical resource to an image list.

Pascal

```
TResType = (
  rtBitmap,
  rtCursor,
  rtIcon
);
```

C++

```
enum TResType {
  rtBitmap,
  rtCursor,
  rtIcon
};
```

File

ImgList

Description

TResTypes is used by the image list methods ResourceLoad, GetResource and FileLoad to specify the type of graphical resource that is to be loaded or retrieved from an image list.

Value	Meaning
rtBitmap	Loads or retrieves a bitmap
rtCursor	Loads or retrieves a cursor
rtIcon	Loads or retrieves an icon

19.3 Types

The following table lists types in this documentation.

Types

Name	Description
TImageIndex (see page 689)	This is type ImgList.TImageIndex.
TImageListHandle (see page 689)	This is type ImgList.TImageListHandle.
TLoadResources (see page 689)	TLoadResource (see page 687) and TLoadResources determine how a resource is loaded into an image list.
TOverlay (see page 690)	TOverlay indicates the index of an overlay mask.

19.3.1 ImgList.TImageIndex Type

Pascal

```
TImageIndex = type Integer;
```

C++

```
type Integer TImageIndex;
```

File

ImgList

Description

This is type ImgList.TImageIndex.

19.3.2 ImgList.TImageListHandle Type

Pascal

```
TImageListHandle = HImageList;
```

C++

```
HImageList TImageListHandle;
```

File

ImgList

Description

This is type ImgList.TImageListHandle.

19.3.3 ImgList.TLoadResources Type

TLoadResource (see page 687) and TLoadResources determine how a resource is loaded into an image list.

Pascal

```
TLoadResources = set of TLoadResource;
```

C++

```
TLoadResource TLoadResources;
```

File

ImgList

Description

TLoadResources is a set of flags (TLoadResource (see page 687) values) that govern the way an image list loads image resources. The following table lists the possible TLoadResource (see page 687) values.

Value	Meaning
lrDefaultColor	Use the color format of the display.
lrDefaultSize	Use system metrics for Height and Width if the Width parameter to GetResource is 0. If the Width parameter is 0 and this option is not specified for cursors and icons, the resources width and height will be used.
lrFromFile	Loads the resource from a specified file.
lrMap3DColors	Replaces the images' gray shades with the 3D windows colors.
lrTransparent	Replaces the color table entry for the first pixel of the resource with the default window color (COLOR_WINDOW). This applies only to images with color tables.
lrMonoChrome	Loads the resource as black and white.

See Also

SelectDirectory

19.3.4 ImgList.TOverlay Type

TOverlay indicates the index of an overlay mask.

Pascal

```
TOverlay = 0..3;
```

C++

```
0..3 TOverlay;
```

File

ImgList

Description

TOverlay has a value between 0 and 3 (inclusive). It indicates the index of an image in an image list that acts as an overlay mask.

20 IMouse Namespace

This is namespace IMouse.

Topics

Name	Description
Classes (see page 693)	The following table lists classes in this documentation.
Functions (see page 706)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 707)	The following table lists structs, records, enums in this documentation.
Types (see page 709)	The following table lists types in this documentation.
Constants (see page 711)	The following table lists constants in this documentation.

Classes

	Name	Description
	TPanningWindow (see page 693)	This is a class of TCustomPanningWindow.

Constants

Name	Description
crPanAll (see page 711)	This is constant IMouse.crPanAll.
crPanDown (see page 711)	This is constant IMouse.crPanDown.
crPanDownLeft (see page 712)	This is constant IMouse.crPanDownLeft.
crPanDownRight (see page 712)	This is constant IMouse.crPanDownRight.
crPanLeft (see page 712)	This is constant IMouse.crPanLeft.
crPanLeftRight (see page 713)	This is constant IMouse.crPanLeftRight.
crPanRight (see page 713)	This is constant IMouse.crPanRight.
crPanUp (see page 713)	This is constant IMouse.crPanUp.
crPanUpDown (see page 714)	This is constant IMouse.crPanUpDown.
crPanUpLeft (see page 714)	This is constant IMouse.crPanUpLeft.
crPanUpRight (see page 714)	This is constant IMouse.crPanUpRight.

Functions

	Name	Description
	StartPan (see page 706)	This is function IMouse.StartPan.

Structs, Records, Enums

	Name	Description
	TPanDirection (see page 707)	The type describes the panning direction.

	TPanOption (see page 707)	This is record IMouse.TPanOption.

Types

Name	Description
TPanDirections (see page 709)	Set of TPanDirection (see page 707). .
TPanOptions (see page 709)	This is type IMouse.TPanOptions.
TUpdateEvent (see page 709)	Initiates the cursor update.

20.1 Classes

The following table lists classes in this documentation.

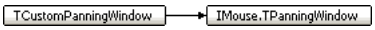
Classes

	Name	Description
	TPanningWindow (see page 693)	This is a class of TCustomPanningWindow.

20.1.1 TPanningWindow Class

This is a class of TCustomPanningWindow.

Class Hierarchy



Pascal

```
TPanningWindow = class(TCustomPanningWindow);
```

C++

```
class TPanningWindow : public TCustomPanningWindow;
```

File

IMouse

Description

Use this type if you are creating a specialized version of TPanningWindowClass.

20.1.1.1 TPanningWindow Members

The following tables list the members exposed by TPanningWindow.

Protected Methods

	Name	Description
	CreateParams (see page 695)	Initializes a window-creation parameter data structure.
	DoUpdate (see page 696)	This is DoUpdate, a member of class TPanningWindow.
	GetPanOptions (see page 696)	This is GetPanOptions, a member of class TPanningWindow.
	MouseMove (see page 697)	Respond to mouse moving over control area..
	Paint (see page 697)	Renders the image of a custom control.
	SetPanCursor (see page 698)	This is SetPanCursor, a member of class TPanningWindow.
	UpdateScroll (see page 698)	This is UpdateScroll, a member of class TPanningWindow.
	WndProc (see page 699)	Provides specific message responses for the control.

Public Events

	Name	Description
	OnUpdate (see page 704)	Initiates the cursor update.

Public Methods

	Name	Description
	Create (see page 694)	Creates an instance of TCustomControl.
	Destroy (see page 694)	Destroys an instance of TCustomControl.
	GetIsPanning (see page 696)	Returns whether or not the mouse is actively panning.
	StartPanning (see page 698)	Tells the panning window to start the panning operation.
	StopPanning (see page 698)	Tells the panning window to stop the panning operation.

Public Properties

	Name	Description
	IsPanning (see page 703)	Returns GetIsPanning (see page 696).
	PanControl (see page 703)	The control that is being panned. .
	PanInterval (see page 704)	The interval that determines when to update the cursor, send events, etc.
	PanOptions (see page 704)	Vertical panning, Horizontal panning, both or neither (it is a set).

20.1.1.2 TPanningWindow.Create Constructor

Creates an instance of TCustomControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TPanningWindow(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomControl. However, you should never attempt to instantiate a TCustomControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants. Controls placed on forms at design time are created automatically.

After calling the inherited constructor, Create initializes the control, including the following property settings:

Initializes a TCanvas object for its Canvas property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Create

Destroy

Canvas

Owner

20.1.1.3 TPanningWindow.Destroy Destructor

Destroys an instance of TCustomControl.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TPanningWindow();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the control is not nil, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

As the control is destroyed, it:

Destroys the TCanvas object in its Canvas property.

Override Destroy to free any memory or resources allocated in the Create method. When declaring a Destroy method in a descendant control type, always add the override directive to the declaration and call the inherited Destroy as the last statement in the redeclared method.

When a control is created, Delphi automatically provides exception handling around the constructor call. The destructor is called if an exception escapes from the constructor. This means that the destructor code must be prepared to clean up a partially constructed instance. Check all data fields for zero before disposing of their contents.

See Also

System::TObject::Free

delete

Create

20.1.1.4 TPanningWindow Methods

The methods of the TPanningWindow class are listed here.

Protected Methods

	Name	Description
	CreateParams (see page 695)	Initializes a window-creation parameter data structure.
	DoUpdate (see page 696)	This is DoUpdate, a member of class TPanningWindow.
	GetPanOptions (see page 696)	This is GetPanOptions, a member of class TPanningWindow.
	MouseMove (see page 697)	Respond to mouse moving over control area..
	Paint (see page 697)	Renders the image of a custom control.
	SetPanCursor (see page 698)	This is SetPanCursor, a member of class TPanningWindow.
	UpdateScroll (see page 698)	This is UpdateScroll, a member of class TPanningWindow.
	WndProc (see page 699)	Provides specific message responses for the control.

Public Methods

	Name	Description
	GetIsPanning (see page 696)	Returns whether or not the mouse is actively panning.
	StartPanning (see page 698)	Tells the panning window to start the panning operation.
	StopPanning (see page 698)	Tells the panning window to stop the panning operation.

20.1.1.4.1 TPanningWindow.CreateParams Method

Initializes a window-creation parameter data structure.

Pascal

```
procedure CreateParams(var Params: TCreateParams); override;
```

C++

```
virtual __fastcall CreateParams(TCreateParams Params);
```

Description

The CreateWnd method calls CreateParams to initialize the parameters it passes to CreateWindowHandle. Override CreateParams to customize the way a control creates its Windows representation. When overriding CreateParams, always call the inherited method first to set the default values, then make any desired adjustments.

CreateParams fills in the parameter that is passed as the value of the Params parameter. This data structure holds information needed when telling Windows to create a window handle. The fields of a TCreateParams record become the parameters to a call to the CreateWindowEx API function.

TWinControl implements CreateParams by setting all the fields of Params to generic base values. A number of the standard controls override CreateParams to change one or more of the default values in Params.

See Also

CreateWindowHandle

CreateWnd

AddBiDiModeExStyle

20.1.1.4.2 TPanningWindow.DoUpdate Method

Pascal

```
procedure DoUpdate(var Delta: TPoint; var CursorDirection: TPanDirections); dynamic;
```

C++

```
__fastcall DoUpdate(TPoint Delta, TPanDirections CursorDirection);
```

Description

This is DoUpdate, a member of class TPanningWindow.

20.1.1.4.3 TPanningWindow.GetIsPanning Method

Returns whether or not the mouse is actively panning.

Pascal

```
function GetIsPanning: Boolean; override;
```

C++

```
virtual __fastcall Boolean GetIsPanning();
```

Description

This boolean value returns True if the mouse is actively panning; false if it is not.

20.1.1.4.4 TPanningWindow.GetPanOptions Method

Pascal

```
function GetPanOptions: TPanOptions; virtual;
```

C++

```
virtual __fastcall TPanOptions GetPanOptions();
```

Description

This is GetPanOptions, a member of class TPanningWindow.

20.1.1.4.5 TPanningWindow.MouseMove Method

Respond to mouse moving over control area..

Pascal

```
procedure MouseMove(Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseMove(TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseMove method to provide other responses in addition to calling the OnMouseMove event handler when the user moves the mouse.

A control calls MouseMove in response to any of the Windows mouse-move messages (WM_MOUSEMOVE), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively.

As the mouse cursor moves across a control, this method is called repeatedly. Each time it is called, it is with the new coordinates that reflect the continuous path of the mouse cursor across the screen real estate covered by the control's visual representation.

See Also

MouseDown

MouseUp

OnMouseMove

20.1.1.4.6 TPanningWindow.Paint Method

Renders the image of a custom control.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method for TCustomControl does nothing other than provide the interface for a method that responds to WM_PAINT messages. Paint is called by the PaintWindow method, after that method has supplied the Canvas with the handle to a device context. When creating a custom control, always override Paint to draw the image of the control.

Tip: To determine which portions of the control's canvas need to be repainted when Paint is called, use the ClipRect property of the canvas.

See Also

Canvas

PaintWindow

GetDeviceContext

ClipRect

20.1.1.4.7 TPanningWindow.SetPanCursor Method

Pascal

```
procedure SetPanCursor(ADirection: TPanDirections); virtual;
```

C++

```
virtual __fastcall SetPanCursor(TPanDirections ADirection);
```

Description

This is SetPanCursor, a member of class TPanningWindow.

20.1.1.4.8 TPanningWindow.StartPanning Method

Tells the panning window to start the panning operation.

Pascal

```
function StartPanning(AHandle: THandle; AControl: TControl): Boolean; override;
```

C++

```
virtual __fastcall Boolean StartPanning(THandle AHandle, TControl * AControl);
```

Description

Use this method to initiate panning.

20.1.1.4.9 TPanningWindow.StopPanning Method

Tells the panning window to stop the panning operation.

Pascal

```
procedure StopPanning; override;
```

C++

```
virtual __fastcall StopPanning();
```

Description

Use this method to cease panning.

20.1.1.4.10 TPanningWindow.UpdateScroll Method

Pascal

```
procedure UpdateScroll(Pt: TPoint); virtual;
```

C++

```
virtual __fastcall UpdateScroll(TPoint Pt);
```

Description

This is UpdateScroll, a member of class TPanningWindow.

20.1.1.4.11 TPanningWindow.WndProc Method

Provides specific message responses for the control.

Pascal

```
procedure WndProc(var Msg: TMessage); override;
```

C++

```
virtual __fastcall WndProc(TMessage Msg);
```

Description

Override WndProc to change the initial Windows message handler for the control. The WindowProc property is initialized to point to the WndProc method.

WndProc for TWinControl overrides the inherited method to define responses for focus, mouse, and keyboard messages. It sends all others to its inherited WndProc.

When overriding WndProc to provide specialized responses to messages, call the inherited WndProc at the end to dispatch any unhandled messages.

C++ Examples:

```
/*
This example shows how to use the WndProc method and the
WindowProc property to subclass a custom control's window
procedure.  This example subclasses the window procedure of
a TListBox descendant to respond to a user-defined message
called WM_STYLEMESSAGE. The subclassed window procedure can
be turned on or off by pressing a radio button.
*/
class TMyListBoxDescendant : public TListBox
{
__published:    // IDE-managed Components
    void __fastcall SubClassWndProc(Messages::TMessage &Message);
    void __fastcall ToggleSubClass(bool On);
    void __fastcall OnDrawItemProc(
      TWinControl *Control, int Index, const TRect &Rect,
      TOwnerDrawState State);
private:    // User declarations
public:     // User declarations
    __fastcall TMyListBoxDescendant(TComponent* Owner);
};

TForm1 *Form1;
TMyListBoxDescendant *MyListBoxDescendant1;
Graphics::TBitmap *bitmap0;

const WM_STYLEMESSAGE = WM_USER + 2000;

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

__fastcall TMyListBoxDescendant::TMyListBoxDescendant(TComponent* Owner)
    : TListBox(Owner)
{
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  PostMessage(
    MyListBoxDescendant1->Handle,
```

```
    WM_STYLEMESSAGE,
    Integer(lbOwnerDrawFixed),
    0);
}

void __fastcall TForm1::Button2Click(TObject *Sender)
{
  PostMessage(
    MyListBoxDescendant1->Handle,
    WM_STYLEMESSAGE,
    Integer(lbStandard),
    0);
}

void __fastcall TForm1::SubClassRadioGroup1Click(TObject *Sender)
{
  MyListBoxDescendant1->ToggleSubClass(SubClassRadioGroup1->ItemIndex == 0);
}

void __fastcall TMyListBoxDescendant::SubClassWndProc(Messages::TMessage &Message)
{
  if (Message.Msg == WM_STYLEMESSAGE)
    Style = (TListBoxStyle)Message.WParam;
  else
    WndProc(Message);
}

void __fastcall TMyListBoxDescendant::ToggleSubClass(bool On)
{
  if (On)
    WindowProc = SubClassWndProc;
  else
    WindowProc = WndProc;
}

#include <memory>       //for STL auto_ptr class

void __fastcall TMyListBoxDescendant::OnDrawItemProc(TWinControl *Control, int Index,
      const TRect &Rect, TOwnerDrawState State)
{
  Graphics::TBitmap *bitmap; // temporary variable for item's bitmap
  int Offset = 2;   // default text offset width
  // note that we draw on the listbox's canvas, not on the form
  TCanvas *canvas = dynamic_cast<TListBox *>(Control)->Canvas;
  canvas->FillRect(Rect); // clear the rectangle
  bitmap = dynamic_cast<Graphics::TBitmap *>((dynamic_cast<TListBox
*>(Control))->Items->Objects[Index]);
  if (bitmap)
  {
    canvas->BrushCopy(
      Bounds(Rect.Left + Offset, Rect.Top, bitmap->Width, bitmap->Height),
      bitmap, Bounds(0, 0, bitmap->Width, bitmap->Height), clRed); // render bitmap
    Offset += bitmap->Width + 4;   // add four pixels between bitmap and text
  }
  // display the text
  canvas->TextOut(Rect.Left + Offset, Rect.Top, dynamic_cast<TListBox
*>(Control)->Items->Strings[Index]);
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  MyListBoxDescendant1 = new TMyListBoxDescendant(Form1); // The owner cleans this up.
  MyListBoxDescendant1->Visible = True;
  MyListBoxDescendant1->Parent = Form1;
  MyListBoxDescendant1->Visible = True;
  MyListBoxDescendant1->Left =
```

```
    SubClassRadioGroup1->Left + SubClassRadioGroup1->Width + 30;;
  MyListBoxDescendant1->Top = SubClassRadioGroup1->Top;
  MyListBoxDescendant1->Height = SubClassRadioGroup1->Height;
  MyListBoxDescendant1->OnDrawItem =
    MyListBoxDescendant1->OnDrawItemProc;

  static std::auto_ptr<Graphics::TBitmap> _bitmap0Cleaner(bitmap0 = new Graphics::TBitmap);
  ImageList1->GetBitmap(0, bitmap0);
  MyListBoxDescendant1->Items->AddObject("Butterfly", bitmap0);

  SubClassRadioGroup1->Items->Add("SubClassWndProc");
  SubClassRadioGroup1->Items->Add("WndProc");
  SubClassRadioGroup1->ItemIndex = 2;
}
```

Delphi Examples:

```
{
This example shows how to use the WndProc method and the
WindowProc property to subclass a custom control's window
procedure.  This example subclasses the window procedure of
a TListBox descendant to respond to a user-defined message
called WM_STYLEMESSAGE. The subclassed window procedure can
be turned on or off by pressing a radio button.
}

type
  TMyListBoxDescendant = class(TlistBox)
    procedure SubClassWndProc(var Message: TMessage);
    procedure ToggleSubClass(On: Boolean);
    procedure OnDrawItemProc(
      Control: TWinControl;
      Index: Integer;
      Rect:TRect;
      State: TOwnerDrawState);
  end;
  TForm1 = class(TForm)
    SubClassRadioGroup1: TRadioGroup;
    Button1: TButton;
    ImageList1: TImageList;
    Button2: TButton;
    procedure SubClassRadioGroup1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  MyListBoxDescendant1: TMyListBoxDescendant;
  bitmap0: TBitmap;

implementation

{$R *.dfm}

const WM_STYLEMESSAGE = WM_USER + 2000;

procedure TForm1.Button1Click(Sender: TObject);
begin
  PostMessage(
    MyListBoxDescendant1.Handle,
```

```
    WM_STYLEMESSAGE,
    Integer(lbOwnerDrawFixed),
    0);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  PostMessage(
    MyListBoxDescendant1.Handle,
    WM_STYLEMESSAGE,
    Integer(lbStandard),
    0);
end;

procedure TMyListBoxDescendant.SubClassWndProc(var Message: TMessage);
begin
  if (Message.Msg = WM_STYLEMESSAGE) then
    Style:= TListBoxStyle(Message.WParam)
  else
    WndProc(Message);
end;

procedure TMyListBoxDescendant.ToggleSubClass(On: Boolean);
begin
  if On then
    WindowProc := SubClassWndProc
  else
    WindowProc := WndProc;
end;

procedure TForm1.SubClassRadioGroup1Click(Sender: TObject);
begin
  MyListBoxDescendant1.ToggleSubClass(
    SubClassRadioGroup1.ItemIndex = 0);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  MyListBoxDescendant1:= TMyListBoxDescendant.Create(self);
  MyListBoxDescendant1.Visible:= True;
  MyListBoxDescendant1.Parent:= Form1;
  MyListBoxDescendant1.Visible:= True;
  MyListBoxDescendant1.Left:=
    SubClassRadioGroup1.Left + SubClassRadioGroup1.Width + 30;;
  MyListBoxDescendant1.Top:= SubClassRadioGroup1.Top;
  MyListBoxDescendant1.Height:= SubClassRadioGroup1.Height;
  MyListBoxDescendant1.OnDrawItem:=
    MyListBoxDescendant1.OnDrawItemProc;

  bitmap0 := TBitmap.Create;
  ImageList1.GetBitmap(0, bitmap0);
  MyListBoxDescendant1.Items.AddObject('Butterfly', bitmap0);

  SubClassRadioGroup1.Items.Add('SubClassWndProc');
  SubClassRadioGroup1.Items.Add('WndProc');
  SubClassRadioGroup1.ItemIndex := 2;
end;

procedure TMyListBoxDescendant.OnDrawItemProc(
  Control: TWinControl;
  Index: Integer;
  Rect:TRect;
  State: TOwnerDrawState);
var
  Bitmap: TBitmap;      { temporary variable for the item's bitmap }
  Offset: Integer;      { text offset width }
```

```
begin
  { draw on control canvas, not on the form }
  with (Control as TListBox).Canvas do
  begin
    FillRect(Rect);       { clear the rectangle }
    Offset := 2;          { provide default offset }
    Bitmap := TBitmap((Control as TListBox).Items.Objects[Index]);  { get the bitmap }
    if Bitmap <> nil then
    begin
      BrushCopy(
        Bounds(Rect.Left + Offset, Rect.Top, Bitmap.Width, Bitmap.Height),
        Bitmap,
        Bounds(0, 0, Bitmap.Width, Bitmap.Height),
        clRed);  {render bitmap}
      Offset := Bitmap.width + 6;    { add four pixels between bitmap and text}
    end;
    TextOut(Rect.Left + Offset, Rect.Top, (Control as TListBox).Items[Index])  { display the
text }
  end;
end;
```

See Also

DefWndProc

MainWndProc

WndProc

WindowProc

20.1.1.5 TPanningWindow Properties

The properties of the TPanningWindow class are listed here.

Public Properties

	Name	Description
R	IsPanning (see page 703)	Returns GetIsPanning (see page 696).
R	PanControl (see page 703)	The control that is being panned. .
	PanInterval (see page 704)	The interval that determines when to update the cursor, send events, etc.
R	PanOptions (see page 704)	Vertical panning, Horizontal panning, both or neither (it is a set).

20.1.1.5.1 TPanningWindow.IsPanning Property

Returns GetIsPanning (see page 696).

Pascal

```
property IsPanning: Boolean;
```

C++

```
__property Boolean IsPanning;
```

Description

This property runs the GetIsPanning (see page 696) method and stores the result.

20.1.1.5.2 TPanningWindow.PanControl Property

The control that is being panned. .

Pascal

```
property PanControl: TControl;
```

C++

```
__property TControl * PanControl;
```

Description

Use this property to store the control that is being panned.

20.1.1.5.3 TPanningWindow.PanInterval Property

The interval that determines when to update the cursor, send events, etc.

Pascal

```
property PanInterval: Integer;
```

C++

```
__property int PanInterval;
```

Description

Use this property to store the number of minutes between updates for the cursor, send events, etc..

20.1.1.5.4 TPanningWindow.PanOptions Property

Vertical panning, Horizontal panning, both or neither (it is a set).

Pascal

```
property PanOptions: TPanOptions;
```

C++

```
__property TPanOptions PanOptions;
```

Description

This property stores the option chosen for panning.

20.1.1.6 TPanningWindow Events

The events of the TPanningWindow class are listed here.

Public Events

	Name	Description
	OnUpdate (see page 704)	Initiates the cursor update.

20.1.1.6.1 TPanningWindow.OnUpdate Event

Initiates the cursor update.

Pascal

```
property OnUpdate: TUpdateEvent;
```

C++

```
__property TUpdateEvent OnUpdate;
```

Description

This event is called when the cursor is to be updated.

20.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	StartPan (see page 706)	This is function IMouse.StartPan.

20.2.1 IMouse.StartPan Function

Pascal

```
function StartPan(WndHandle: THandle; AControl: TControl): Boolean;
```

C++

```
Boolean StartPan(THandle WndHandle, TControl * AControl);
```

File

IMouse

Description

This is function IMouse.StartPan.

20.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TPanDirection (see page 707)	The type describes the panning direction.
	TPanOption (see page 707)	This is record IMouse.TPanOption.

20.3.1 IMouse.TPanDirection Enumeration

The type describes the panning direction.

Pascal

```
TPanDirection = (
  pdUp,
  pdDown,
  pdLeft,
  pdRight
);
```

C++

```
enum TPanDirection {
  pdUp,
  pdDown,
  pdLeft,
  pdRight
};
```

File

IMouse

Description

The possible panning values are: pdUp, pdDown, pdLeft, pdRight.

20.3.2 IMouse.TPanOption Enumeration

Pascal

```
TPanOption = (
  poVertical,
  poHorizontal
);
```

C++

```
enum TPanOption {
  poVertical,
  poHorizontal
};
```

File

IMouse

Description

This is record IMouse.TPanOption.

20.4 Types

The following table lists types in this documentation.

Types

Name	Description
TPanDirections (see page 709)	Set of TPanDirection (see page 707). .
TPanOptions (see page 709)	This is type IMouse.TPanOptions.
TUpdateEvent (see page 709)	Initiates the cursor update.

20.4.1 IMouse.TPanDirections Type

Set of TPanDirection (see page 707). .

Pascal

```
TPanDirections = set of TPanDirection;
```

C++

```
TPanDirection TPanDirections;
```

File

IMouse

Description

Set of TPanDirection (see page 707).

20.4.2 IMouse.TPanOptions Type

Pascal

```
TPanOptions = set of TPanOption;
```

C++

```
TPanOption TPanOptions;
```

File

IMouse

Description

This is type IMouse.TPanOptions.

20.4.3 IMouse.TUpdateEvent Type

Initiates the cursor update.

Pascal

```
TUpdateEvent = procedure (Sender: TObject; var Delta: TPoint; var CursorDirection:
TPanDirections) of object;
```

C++

```
(Sender: TObject; var Delta: TPoint; var CursorDirection: TPanDirections) ( TUpdateEvent)();
```

File

IMouse

Description

This event is called when the cursor is to be updated.

See Also

Procedural types

Creating events

20.5 Constants

The following table lists constants in this documentation.

Constants

Name	Description
crPanAll (see page 711)	This is constant IMouse.crPanAll.
crPanDown (see page 711)	This is constant IMouse.crPanDown.
crPanDownLeft (see page 712)	This is constant IMouse.crPanDownLeft.
crPanDownRight (see page 712)	This is constant IMouse.crPanDownRight.
crPanLeft (see page 712)	This is constant IMouse.crPanLeft.
crPanLeftRight (see page 713)	This is constant IMouse.crPanLeftRight.
crPanRight (see page 713)	This is constant IMouse.crPanRight.
crPanUp (see page 713)	This is constant IMouse.crPanUp.
crPanUpDown (see page 714)	This is constant IMouse.crPanUpDown.
crPanUpLeft (see page 714)	This is constant IMouse.crPanUpLeft.
crPanUpRight (see page 714)	This is constant IMouse.crPanUpRight.

20.5.1 IMouse.crPanAll Constant

Pascal

```
crPanAll = TCursor(-50);
```

C++

```
crPanAll = TCursor(-50);
```

File

IMouse

Description

This is constant IMouse.crPanAll.

20.5.2 IMouse.crPanDown Constant

Pascal

```
crPanDown = TCursor(-51);
```

C++

```
crPanDown = TCursor(-51);
```

File

IMouse

Description

This is constant IMouse.crPanDown.

20.5.3 IMouse.crPanDownLeft Constant

Pascal

```
crPanDownLeft = TCursor(-52);
```

C++

```
crPanDownLeft = TCursor(-52);
```

File

IMouse

Description

This is constant IMouse.crPanDownLeft.

20.5.4 IMouse.crPanDownRight Constant

Pascal

```
crPanDownRight = TCursor(-53);
```

C++

```
crPanDownRight = TCursor(-53);
```

File

IMouse

Description

This is constant IMouse.crPanDownRight.

20.5.5 IMouse.crPanLeft Constant

Pascal

```
crPanLeft = TCursor(-54);
```

C++

```
crPanLeft = TCursor(-54);
```

File

IMouse

Description

This is constant IMouse.crPanLeft.

20.5.6 IMouse.crPanLeftRight Constant

Pascal

```
crPanLeftRight = TCursor(-55);
```

C++

```
crPanLeftRight = TCursor(-55);
```

File

IMouse

Description

This is constant IMouse.crPanLeftRight.

20.5.7 IMouse.crPanRight Constant

Pascal

```
crPanRight = TCursor(-56);
```

C++

```
crPanRight = TCursor(-56);
```

File

IMouse

Description

This is constant IMouse.crPanRight.

20.5.8 IMouse.crPanUp Constant

Pascal

```
crPanUp = TCursor(-57);
```

C++

```
crPanUp = TCursor(-57);
```

File

IMouse

Description

This is constant IMouse.crPanUp.

20.5.9 IMouse.crPanUpDown Constant

Pascal

```
crPanUpDown = TCursor(-58);
```

C++

```
crPanUpDown = TCursor(-58);
```

File

IMouse

Description

This is constant IMouse.crPanUpDown.

20.5.10 IMouse.crPanUpLeft Constant

Pascal

```
crPanUpLeft = TCursor(-59);
```

C++

```
crPanUpLeft = TCursor(-59);
```

File

IMouse

Description

This is constant IMouse.crPanUpLeft.

20.5.11 IMouse.crPanUpRight Constant

Pascal

```
crPanUpRight = TCursor(-60);
```

C++

```
crPanUpRight = TCursor(-60);
```

File

IMouse

Description

This is constant IMouse.crPanUpRight.

21 IniFiles Namespace

This is namespace IniFiles.

Topics

Name	Description
Classes (see page 716)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 787)	The following table lists structs, records, enums in this documentation.
Types (see page 788)	The following table lists types in this documentation.

Classes

	Name	Description
	EIniFileException (see page 716)	EIniFileException is the exception class for errors that occur when working with INI files using TIniFile (see page 752).
	TCustomIniFile (see page 716)	TCustomIniFile is the base class for components that encapsulate access to .ini files and the Windows system registry.
	THashedStringList (see page 749)	THashedStringList maintains a list of strings using an internal hash table.
	TIniFile (see page 752)	TIniFile stores and retrieves application-specific information and settings from INI files.
	TMemIniFile (see page 765)	TMemIniFile enables buffered storage and retrieval of application-specific information and settings in an INI file.
	TStringHash (see page 783)	This is class IniFiles.TStringHash.

Structs, Records, Enums

	Name	Description
	THashItem (see page 787)	This is record IniFiles.THashItem.

Types

Name	Description
PHashItem (see page 788)	This is type IniFiles.PHashItem.
PPHashItem (see page 788)	This is type IniFiles.PPHashItem.

21.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	EIniFileException (see page 716)	EIniFileException is the exception class for errors that occur when working with INI files using TIniFile (see page 752).
	TCustomIniFile (see page 716)	TCustomIniFile is the base class for components that encapsulate access to .ini files and the Windows system registry.
	THashedStringList (see page 749)	THashedStringList maintains a list of strings using an internal hash table.
	TIniFile (see page 752)	TIniFile stores and retrieves application-specific information and settings from INI files.
	TMemIniFile (see page 765)	TMemIniFile enables buffered storage and retrieval of application-specific information and settings in an INI file.
	TStringHash (see page 783)	This is class IniFiles.TStringHash.

21.1.1 EIniFileException Class

EIniFileException is the exception class for errors that occur when working with INI files using TIniFile (see page 752).

Class Hierarchy

Pascal

```
EIniFileException = class(Exception);
```

C++

```
class EIniFileException : public Exception;
```

File

IniFiles

Description

EIniFileException is raised when TIniFile (see page 752) fails in an attempt to modify its associated INI file.

See Also

TIniFile (see page 752)

21.1.2 TCustomIniFile Class

TCustomIniFile is the base class for components that encapsulate access to .ini files and the Windows system registry.

Class Hierarchy

Pascal

```
TCustomIniFile = class(TObject);
```

C++

```
class TCustomIniFile : public TObject;
```

File

IniFiles

Description

Note: Use TCustomIniFile as a base class when defining components that access ini files. Ini files are text files that are divided into sections with variable assignments in each section. TCustomIniFile introduces properties and methods to handle storage and retrieval of application-specific information and settings in a standard ini file.

Note: On Windows, TCustomIniFile is also the base class for TRegistryIniFile, which allows applications that use both ini files and the system registry to handle both media in a unified manner. This differs from the features of TRegistry, which corresponds more directly to Windows system registry native semantics.

Do not create instances of TCustomIniFile. Instead, use or create descendants of TCustomIniFile, such as TIniFile (see page 752), TRegistryIniFile, and TMemIniFile (see page 765).

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
```

```
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
```

```
        false: WindowState := wsNormal;
      end;

    finally
      SettingsFile.Free;
    end;
  end;

  procedure TForm2.btStoreClick(Sender: TObject);
  var
    SettingsFile: TCustomIniFile;
  begin
    { Open an instance }
    SettingsFile := OpenIniFileInstance();

    {
    Store current form properties to be used in later sessions.
    }
    try
      SettingsFile.WriteInteger (Name, 'Top', Top);
      SettingsFile.WriteInteger (Name, 'Left', Left);
      SettingsFile.WriteInteger (Name, 'Width', Width);
      SettingsFile.WriteInteger (Name, 'Height', Height);
      SettingsFile.WriteString  (Name, 'Caption', Caption);
      SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
      SettingsFile.WriteDateTime(Name, 'LastRun', Now);
    finally
      SettingsFile.Free;
    end;

  end;

  function TForm2.OpenIniFileInstance: TCustomIniFile;
  begin
    {
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    }

    case RadioGroup1.ItemIndex of
      0:
        begin
          { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
          Result := TRegistryIniFile.Create('Software\' + Application.Title);
        end;
      1:
        begin
          { Ini file mode selected }
          Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
      2:
        begin
          { Memory based Ini file mode selected }
          Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
    end;
  end;
```

See Also

TIniFile (see page 752)

TMemIniFile (see page 765)

TRegistry (see page 3246)

TRegistryIniFile (see page 3299)

21.1.2.1 TCustomIniFile Members

The following tables list the members exposed by TCustomIniFile.

Public Methods

	Name	Description
	Create (see page 720)	Creates a TCustomIniFile (see page 716) object for an application.
	DeleteKey (see page 721)	Erases the data value associated with an ini file entry.
	EraseSection (see page 722)	Erases an entire section of an ini file.
	ReadBinaryStream (see page 722)	Retrieves a binary value from the ini file.
	ReadBool (see page 723)	Retrieves a Boolean value from an ini file.
	ReadDate (see page 726)	Retrieves a date value from an ini file.
	ReadDateTime (see page 726)	Retrieves a date-time value from an ini file.
	ReadFloat (see page 727)	Retrieves a float value from an ini file.
	ReadInteger (see page 727)	Retrieves an integer value from an ini file.
	ReadSection (see page 730)	Reads all the key names from a specified section of an ini file into a string list.
	ReadSections (see page 731)	This is the overview for the ReadSections method overload.
	ReadSectionValues (see page 732)	Reads the values from all keys in a section of an ini file into a string list.
	ReadString (see page 732)	Retrieves a string value from an ini file.
	ReadTime (see page 735)	Retrieves a time value from an ini file.
	SectionExists (see page 736)	Indicates whether a section exists in the ini file.
	UpdateFile (see page 736)	Flushes buffered ini file data to disk.
	ValueExists (see page 736)	Indicates whether a key exists in the ini file.
	WriteBinaryStream (see page 737)	Writes a binary value to the ini file.
	WriteBool (see page 737)	Writes a Boolean value to an ini file.
	WriteDate (see page 741)	Writes a date value to an ini file.
	WriteDateTime (see page 741)	Writes a date-time value to an ini file.
	WriteFloat (see page 742)	Writes a float value to an ini file.
	WriteInteger (see page 742)	Writes an integer value to an ini file.
	WriteString (see page 745)	Writes a string value to an ini file.
	WriteTime (see page 748)	Writes a time value to an ini file.

Public Properties

	Name	Description
	FileName (see page 749)	Contains the name of the ini file from which to read and to which to write information.

21.1.2.2 TCustomIniFile.Create Constructor

Creates a TCustomIniFile (see page 716) object for an application.

Pascal

```
constructor Create(const FileName: string);
```

C++

```
__fastcall TCustomIniFile(const UnicodeString FileName);
```

Description

Create constructs a TCustomIniFile (see page 716) object for an application. Create assigns the FileName (see page 749) parameter to the FileName (see page 749) property, which is used to specify the name of the ini file to use.

See Also

FileName (see page 749)

System::TObject::Create

21.1.2.3 TCustomIniFile Methods

The methods of the TCustomIniFile class are listed here.

Public Methods

	Name	Description
	DeleteKey (see page 721)	Erases the data value associated with an ini file entry.
	EraseSection (see page 722)	Erases an entire section of an ini file.
	ReadBinaryStream (see page 722)	Retrieves a binary value from the ini file.
	ReadBool (see page 723)	Retrieves a Boolean value from an ini file.
	ReadDate (see page 726)	Retrieves a date value from an ini file.
	ReadDateTime (see page 726)	Retrieves a date-time value from an ini file.
	ReadFloat (see page 727)	Retrieves a float value from an ini file.
	ReadInteger (see page 727)	Retrieves an integer value from an ini file.
	ReadSection (see page 730)	Reads all the key names from a specified section of an ini file into a string list.
	ReadSections (see page 731)	This is the overview for the ReadSections method overload.
	ReadSectionValues (see page 732)	Reads the values from all keys in a section of an ini file into a string list.
	ReadString (see page 732)	Retrieves a string value from an ini file.
	ReadTime (see page 735)	Retrieves a time value from an ini file.
	SectionExists (see page 736)	Indicates whether a section exists in the ini file.
	UpdateFile (see page 736)	Flushes buffered ini file data to disk.
	ValueExists (see page 736)	Indicates whether a key exists in the ini file.
	WriteBinaryStream (see page 737)	Writes a binary value to the ini file.
	WriteBool (see page 737)	Writes a Boolean value to an ini file.
	WriteDate (see page 741)	Writes a date value to an ini file.
	WriteDateTime (see page 741)	Writes a date-time value to an ini file.
	WriteFloat (see page 742)	Writes a float value to an ini file.
	WriteInteger (see page 742)	Writes an integer value to an ini file.
	WriteString (see page 745)	Writes a string value to an ini file.
	WriteTime (see page 748)	Writes a time value to an ini file.

21.1.2.3.1 TCustomIniFile.DeleteKey Method

Erases the data value associated with an ini file entry.

Pascal

```
procedure DeleteKey(const Section: String; const Ident: String); virtual; abstract;
```

C++

```
virtual __fastcall DeleteKey(const UnicodeString Section, const UnicodeString Ident) = 0;
```

Description

Call DeleteKey to erase a data value associated with an ini file entry. Section is string containing the name of an ini file section, and Ident is a string containing the name of the key for which to set a nil (Delphi) or NULL (C++) value.

Note: Attempting to erase a data value in a nonexistent section or attempting to erase data from a nonexistent key are not errors. In these cases, DeleteKey creates the section and key and sets its initial value to an empty value.

See Also

EraseSection (see page 722)

21.1.2.3.2 TCustomIniFile.EraseSection Method

Erases an entire section of an ini file.

Pascal

```
procedure EraseSection(const Section: string); virtual; abstract;
```

C++

```
virtual __fastcall EraseSection(const UnicodeString Section) = 0;
```

Description

Call EraseSection to remove a section, all its keys, and their data values from an ini file. Section identifies the ini file section to remove. If a section cannot be removed, an exception is raised.

See Also

SectionExists (see page 736)

DeleteKey (see page 721)

21.1.2.3.3 TCustomIniFile.ReadBinaryStream Method

Retrieves a binary value from the ini file.

Pascal

```
function ReadBinaryStream(const Section: string; const Name: string; Value: TStream): Integer;
virtual;
```

C++

```
virtual __fastcall int ReadBinaryStream(const UnicodeString Section, const UnicodeString Name,
TStream Value);
```

Description

Call ReadBinaryStream to read a binary value into a specified stream. After reading the binary value, which it converts from a series of hexadecimal characters, from the ini file into the current position of the stream, ReadBinaryStream returns the number of bytes written to the stream. If the stream is a memory stream (TMemoryStream or one of its descendants), the stream is left positioned to the same byte as it was before the call to ReadBinaryStream (immediately before the new binary value).

Section is the section that contains the desired key.

Name is the key under which the binary data is stored.

Value is a stream to which the binary value is written.

Note: When used with a Windows .ini file, ReadBinaryStream is limited to 1023 characters,

See Also

ReadString (see page 732)

WriteBinaryStream (see page 737)

21.1.2.3.4 TCustomIniFile.ReadBool Method

Retrieves a Boolean value from an ini file.

Pascal

```
function ReadBool(const Section: string; const Ident: string; Default: Boolean): Boolean;
virtual;
```

C++

```
virtual __fastcall Boolean ReadBool(const UnicodeString Section, const UnicodeString Ident,
Boolean Default);
```

Description

Call ReadBool to read a Boolean value from an ini file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the Boolean value. Default is the Boolean value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
```

```
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
```

```
        false: WindowState := wsNormal;
      end;

    finally
      SettingsFile.Free;
    end;
  end;

  procedure TForm2.btStoreClick(Sender: TObject);
  var
    SettingsFile: TCustomIniFile;
  begin
    { Open an instance }
    SettingsFile := OpenIniFileInstance();

    {
    Store current form properties to be used in later sessions.
    }
    try
      SettingsFile.WriteInteger (Name, 'Top', Top);
      SettingsFile.WriteInteger (Name, 'Left', Left);
      SettingsFile.WriteInteger (Name, 'Width', Width);
      SettingsFile.WriteInteger (Name, 'Height', Height);
      SettingsFile.WriteString  (Name, 'Caption', Caption);
      SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
      SettingsFile.WriteDateTime(Name, 'LastRun', Now);
    finally
      SettingsFile.Free;
    end;

  end;

  function TForm2.OpenIniFileInstance: TCustomIniFile;
  begin
    {
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    }

    case RadioGroup1.ItemIndex of
      0:
        begin
          { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
          Result := TRegistryIniFile.Create('Software\' + Application.Title);
        end;
      1:
        begin
          { Ini file mode selected }
          Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
      2:
        begin
          { Memory based Ini file mode selected }
          Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
    end;
  end;
```

See Also

ReadDate (see page 726)

ReadDateTime (see page 726)

ReadFloat (see page 727)

ReadInteger (see page 727)

ReadString (see page 732)

ReadTime (see page 735)

21.1.2.3.5 TCustomIniFile.ReadDate Method

Retrieves a date value from an ini file.

Pascal

```
function ReadDate(const Section: string; const Name: string; Default: TDateTime): TDateTime;
virtual;
```

C++

```
virtual __fastcall TDateTime ReadDate(const UnicodeString Section, const UnicodeString Name,
TDateTime Default);
```

Description

Call ReadDate to read a date value from an ini file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the date value. Default is the date value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

See Also

ReadBool (see page 723)

ReadDateTime (see page 726)

ReadFloat (see page 727)

ReadInteger (see page 727)

ReadString (see page 732)

ReadTime (see page 735)

21.1.2.3.6 TCustomIniFile.ReadDateTime Method

Retrieves a date-time value from an ini file.

Pascal

```
function ReadDateTime(const Section: string; const Name: string; Default: TDateTime):
TDateTime; virtual;
```

C++

```
virtual __fastcall TDateTime ReadDateTime(const UnicodeString Section, const UnicodeString
Name, TDateTime Default);
```

Description

Call ReadDateTime to read a date-time value from an ini file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the date-time value. Default is the date-time value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

See Also

ReadBool (see page 723)

ReadDate (see page 726)

ReadFloat (see page 727)

ReadInteger (see page 727)

ReadString (see page 732)

ReadTime (see page 735)

21.1.2.3.7 TCustomIniFile.ReadFloat Method

Retrieves a float value from an ini file.

Pascal

```
function ReadFloat(const Section: string; const Name: string; Default: Double): Double;
virtual;
```

C++

```
virtual __fastcall Double ReadFloat(const UnicodeString Section, const UnicodeString Name,
Double Default);
```

Description

Call ReadFloat to read a float value from an ini file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the float value. Default is the float value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

See Also

ReadBool (see page 723)

ReadDate (see page 726)

ReadDateTime (see page 726)

ReadInteger (see page 727)

ReadString (see page 732)

ReadTime (see page 735)

21.1.2.3.8 TCustomIniFile.ReadInteger Method

Retrieves an integer value from an ini file.

Pascal

```
function ReadInteger(const Section: string; const Ident: string; Default: Longint): Longint;
virtual;
```

C++

```
virtual __fastcall Longint ReadInteger(const UnicodeString Section, const UnicodeString Ident,
Longint Default);
```

Description

Call ReadInteger to read an integer value from an ini file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the value. Default is the integer value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }
```

```
    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();
```

```
    {
    Store current form properties to be used in later sessions.
    }
    try
      SettingsFile.WriteInteger (Name, 'Top', Top);
      SettingsFile.WriteInteger (Name, 'Left', Left);
      SettingsFile.WriteInteger (Name, 'Width', Width);
      SettingsFile.WriteInteger (Name, 'Height', Height);
      SettingsFile.WriteString  (Name, 'Caption', Caption);
      SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
      SettingsFile.WriteDateTime(Name, 'LastRun', Now);
    finally
      SettingsFile.Free;
    end;

  end;

  function TForm2.OpenIniFileInstance: TCustomIniFile;
  begin
    {
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    }

    case RadioGroup1.ItemIndex of
      0:
        begin
          { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
          Result := TRegistryIniFile.Create('Software\' + Application.Title);
        end;
      1:
        begin
          { Ini file mode selected }
          Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
      2:
        begin
          { Memory based Ini file mode selected }
          Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
    end;
  end;
```

See Also

ReadBool (see page 723)

ReadDate (see page 726)

ReadDateTime (see page 726)

ReadFloat (see page 727)

ReadString (see page 732)

ReadTime (see page 735)

21.1.2.3.9 TCustomIniFile.ReadSection Method

Reads all the key names from a specified section of an ini file into a string list.

Pascal

```
procedure ReadSection(const Section: string; Strings: TStrings); virtual; abstract;
```

C++

```
virtual __fastcall ReadSection(const UnicodeString Section, TStrings Strings) = 0;
```

Description

Call ReadSection to retrieve the names of all keys within a specified section of an ini file into a string list object.

Section identifies the section from which to retrieve a list of key names. Strings specifies the string object to hold the retrieved names. Strings can point to a TStrings descendant such as a string list, or to a component property, such as Items for a TListBox component.

See Also

SectionExists (see page 736)

ReadSections (see page 731)

ReadSectionValues (see page 732)

21.1.2.3.10 ReadSections Method

This is the overview for the ReadSections method overload.

Overload List

	Name	Description
	TCustomIniFile.ReadSections (string, TStrings) (see page 731)	Reads the names of all sections in an ini file into a string list.
	TCustomIniFile.ReadSections (TStrings) (see page 731)	Reads the names of all sections in an ini file into a string list.

21.1.2.3.10.1 TCustomIniFile.ReadSections Method (TStrings)

Reads the names of all sections in an ini file into a string list.

Pascal

```
procedure ReadSections(Strings: TStrings); virtual; abstract; overload;
procedure ReadSections(const Section: string; Strings: TStrings); virtual; overload;
```

C++

```
virtual __fastcall ReadSections(TStrings Strings) = 0;
virtual __fastcall ReadSections(const UnicodeString Section, TStrings Strings);
```

Description

Call ReadSections to retrieve the names of all sections in an ini file into a string list.

`Strings` specifies the string object to hold the retrieved names. Strings can point to a TStrings descendant, or to a component property, such as Items for a TListBox component.

`Section` identifies the section from which to retrieve a list of key names.

See Also

ReadSection (see page 730)

ReadSectionValues (see page 732)

21.1.2.3.11 TCustomIniFile.ReadSectionValues Method

Reads the values from all keys in a section of an ini file into a string list.

Pascal

```
procedure ReadSectionValues(const Section: string; Strings: TStrings); virtual; abstract;
```

C++

```
virtual __fastcall ReadSectionValues(const UnicodeString Section, TStrings Strings) = 0;
```

Description

Call ReadSectionValues to read the values from all keys in a specified section of an ini file into a string list object. The values are added to the string list as strings of the form KeyName=ValueName.

Section identifies the section in the file from which to read key values.

Strings is the string list object into which to read the values. Strings can point to a TStrings object, or to a component property, such as Items for a TListBox component.

See Also

SectionExists (see page 736)

ReadSection (see page 730)

ReadSections (see page 731)

21.1.2.3.12 TCustomIniFile.ReadString Method

Retrieves a string value from an ini file.

Pascal

```
function ReadString(const Section: string; const Ident: string; const Default: string):
string; virtual; abstract;
```

C++

```
virtual __fastcall UnicodeString ReadString(const UnicodeString Section, const UnicodeString
Ident, const UnicodeString Default) = 0;
```

Description

Call ReadString to read a string value from a .ini file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the value. Default is the string value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

Note: On Windows, if the string in the .ini file is surrounded by quotation marks, the return value has those marks stripped away. On Linux, the quotation marks are preserved.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
```

```
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
```

```
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
    SettingsFile.WriteInteger (Name, 'Height', Height);
    SettingsFile.WriteString  (Name, 'Caption', Caption);
    SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
    SettingsFile.WriteDateTime(Name, 'LastRun', Now);
  finally
    SettingsFile.Free;
  end;

end;
```

```
function TForm2.OpenIniFileInstance: TCustomIniFile;
begin
  {
  Open/create a new INI file that has the same name as our executable
  only with the INI extension.
  }

  case RadioGroup1.ItemIndex of
    0:
      begin
        { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
        Result := TRegistryIniFile.Create('Software\' + Application.Title);
      end;
    1:
      begin
        { Ini file mode selected }
        Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
    2:
      begin
        { Memory based Ini file mode selected }
        Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
  end;
end;
```

See Also

ReadBool (see page 723)

ReadDate (see page 726)

ReadDateTime (see page 726)

ReadFloat (see page 727)

ReadInteger (see page 727)

ReadTime (see page 735)

ReadBinaryStream (see page 722)

21.1.2.3.13 TCustomIniFile.ReadTime Method

Retrieves a time value from an ini file.

Pascal

```
function ReadTime(const Section: string; const Name: string; Default: TDateTime): TDateTime;
virtual;
```

C++

```
virtual __fastcall TDateTime ReadTime(const UnicodeString Section, const UnicodeString Name,
TDateTime Default);
```

Description

Call ReadTime to read a time value from an ini file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the time value. Default is the time value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

See Also

ReadBool (see page 723)

ReadDate (see page 726)

ReadDateTime (see page 726)

ReadFloat (see page 727)

ReadInteger (see page 727)

ReadString (see page 732)

21.1.2.3.14 TCustomIniFile.SectionExists Method

Indicates whether a section exists in the ini file.

Pascal

```
function SectionExists(const Section: string): Boolean;
```

C++

```
__fastcall Boolean SectionExists(const UnicodeString Section);
```

Description

Use SectionExists to determine whether a section exists within the ini file specified in FileName (see page 749).

Section is the ini file section SectionExists determines the existence of.

SectionExists returns a Boolean value that indicates whether the section in question exists.

See Also

FileName (see page 749)

ValueExists (see page 736)

21.1.2.3.15 TCustomIniFile.UpdateFile Method

Flushes buffered ini file data to disk.

Pascal

```
procedure UpdateFile; virtual; abstract;
```

C++

```
virtual __fastcall UpdateFile() = 0;
```

Description

Call UpdateFile to flush buffered reads from and writes to the ini file to disk. [SK comment -- UpdateFile is useful under Windows95but has no effect under Windows NT as NT does not buffer ini file reads and writes. do not delete!]

See Also

FileName (see page 749)

21.1.2.3.16 TCustomIniFile.ValueExists Method

Indicates whether a key exists in the ini file.

Pascal

```
function ValueExists(const Section: string; const Ident: string): Boolean; virtual;
```

C++

```
virtual __fastcall Boolean ValueExists(const UnicodeString Section, const UnicodeString Ident);
```

Description

Use ValueExists to determine whether a key exists in the ini file specified in FileName (see page 749).

Section is the section in the ini file in which to search for the key.

Ident is the name of the key to search for.

ValueExists returns a boolean value that indicates whether the key exists in the specified section.

See Also

FileName (see page 749)

SectionExists (see page 736)

21.1.2.3.17 TCustomIniFile.WriteBinaryStream Method

Writes a binary value to the ini file.

Pascal

```
procedure WriteBinaryStream(const Section: string; const Name: string; Value: TStream);
virtual;
```

C++

```
virtual __fastcall WriteBinaryStream(const UnicodeString Section, const UnicodeString Name,
TStream Value);
```

Description

Call WriteBinaryStream to write a binary value from a specified stream to a specified section and key in the ini file. WriteBinaryStream writes from the current position in the stream to the end, converting the value into its hexadecimal representation.

Section is the section that contains the target key.

Name is the key under which the binary data is to be written.

Value is a stream to from which the binary value can be read.

Note: Attempting to write a value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteBinaryStream creates the section and key and sets its initial value to the hexadecimal representation of the stream's data.

See Also

WriteString (see page 745)

ReadBinaryStream (see page 722)

21.1.2.3.18 TCustomIniFile.WriteBool Method

Writes a Boolean value to an ini file.

Pascal

```
procedure WriteBool(const Section: string; const Ident: string; Value: Boolean); virtual;
```

C++

```
virtual __fastcall WriteBool(const UnicodeString Section, const UnicodeString Ident, Boolean
Value);
```

Description

Call WriteBool to write a Boolean value to an ini file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the Boolean value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteBool creates the section and key and sets its initial value to Value.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
```

```
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;
```

```
procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
    SettingsFile.WriteInteger (Name, 'Height', Height);
    SettingsFile.WriteString  (Name, 'Caption', Caption);
    SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
    SettingsFile.WriteDateTime(Name, 'LastRun', Now);
  finally
    SettingsFile.Free;
  end;

end;

function TForm2.OpenIniFileInstance: TCustomIniFile;
begin
  {
  Open/create a new INI file that has the same name as our executable
  only with the INI extension.
  }

  case RadioGroup1.ItemIndex of
    0:
      begin
        { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
        Result := TRegistryIniFile.Create('Software\' + Application.Title);
      end;
    1:
      begin
        { Ini file mode selected }
        Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
    2:
      begin
        { Memory based Ini file mode selected }
        Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
  end;
end;
```

See Also

WriteDate (see page 741)

WriteDateTime (see page 741)

WriteFloat (see page 742)

WriteInteger (see page 742)

WriteString (see page 745)

WriteTime (see page 748)

21.1.2.3.19 TCustomIniFile.WriteDate Method

Writes a date value to an ini file.

Pascal

```
procedure WriteDate(const Section: string; const Name: string; Value: TDateTime); virtual;
```

C++

```
virtual __fastcall WriteDate(const UnicodeString Section, const UnicodeString Name, TDateTime
Value);
```

Description

Call WriteDate to write a date value to an ini file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the date value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteDate creates the section and key and sets its initial value to Value.

See Also

WriteBool (see page 737)

WriteDateTime (see page 741)

WriteFloat (see page 742)

WriteInteger (see page 742)

WriteString (see page 745)

WriteTime (see page 748)

21.1.2.3.20 TCustomIniFile.WriteDateTime Method

Writes a date-time value to an ini file.

Pascal

```
procedure WriteDateTime(const Section: string; const Name: string; Value: TDateTime); virtual;
```

C++

```
virtual __fastcall WriteDateTime(const UnicodeString Section, const UnicodeString Name,
TDateTime Value);
```

Description

Call WriteDateTime to write a date-time value to an ini file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the date-time value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteDateTime creates the section and key and sets its initial value to Value.

See Also

WriteBool (see page 737)

WriteDate (see page 741)

WriteFloat (see page 742)

WriteInteger (see page 742)

WriteString (see page 745)

WriteTime (see page 748)

21.1.2.3.21 TCustomIniFile.WriteFloat Method

Writes a float value to an ini file.

Pascal

```
procedure WriteFloat(const Section: string; const Name: string; Value: Double); virtual;
```

C++

```
virtual __fastcall WriteFloat(const UnicodeString Section, const UnicodeString Name, Double
Value);
```

Description

Call WriteFloat to write a float value to an ini file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the float value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteFloat creates the section and key and sets its initial value to Value.

See Also

WriteBool (see page 737)

WriteDate (see page 741)

WriteDateTime (see page 741)

WriteInteger (see page 742)

WriteString (see page 745)

WriteTime (see page 748)

21.1.2.3.22 TCustomIniFile.WriteInteger Method

Writes an integer value to an ini file.

Pascal

```
procedure WriteInteger(const Section: string; const Ident: string; Value: Longint); virtual;
```

C++

```
virtual __fastcall WriteInteger(const UnicodeString Section, const UnicodeString Ident,
Longint Value);
```

Description

Call WriteInteger to write an integer value to an ini file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the integer value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteInteger creates the section and key and sets its initial value to Value.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
```

```
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
```

```
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
    }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
    SettingsFile.WriteInteger (Name, 'Height', Height);
    SettingsFile.WriteString  (Name, 'Caption', Caption);
    SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
    SettingsFile.WriteDateTime(Name, 'LastRun', Now);
  finally
    SettingsFile.Free;
  end;
```

```
end;

function TForm2.OpenIniFileInstance: TCustomIniFile;
begin
  {
  Open/create a new INI file that has the same name as our executable
  only with the INI extension.
  }

  case RadioGroup1.ItemIndex of
    0:
      begin
        { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
        Result := TRegistryIniFile.Create('Software\' + Application.Title);
      end;
    1:
      begin
        { Ini file mode selected }
        Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
    2:
      begin
        { Memory based Ini file mode selected }
        Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
  end;
end;
```

See Also

WriteBool (see page 737)

WriteDate (see page 741)

WriteDateTime (see page 741)

WriteFloat (see page 742)

WriteString (see page 745)

WriteTime (see page 748)

21.1.2.3.23 TCustomIniFile.WriteString Method

Writes a string value to an ini file.

Pascal

```
procedure WriteString(const Section: String; const Ident: String; const Value: String);
virtual; abstract;
```

C++

```
virtual __fastcall WriteString(const UnicodeString Section, const UnicodeString Ident, const
UnicodeString Value) = 0;
```

Description

Call WriteString to write a string value to an ini file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the string value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteString creates the section and key and sets its initial value to Value.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */
```

```
    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
    SettingsFile.WriteInteger (Name, 'Height', Height);
    SettingsFile.WriteString  (Name, 'Caption', Caption);
    SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
```

```
      SettingsFile.WriteDateTime(Name, 'LastRun', Now);
    finally
      SettingsFile.Free;
    end;

  end;

  function TForm2.OpenIniFileInstance: TCustomIniFile;
  begin
    {
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    }

    case RadioGroup1.ItemIndex of
      0:
        begin
          { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
          Result := TRegistryIniFile.Create('Software\' + Application.Title);
        end;
      1:
        begin
          { Ini file mode selected }
          Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
      2:
        begin
          { Memory based Ini file mode selected }
          Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
    end;
  end;
```

See Also

WriteBool (see page 737)

WriteDate (see page 741)

WriteDateTime (see page 741)

WriteFloat (see page 742)

WriteInteger (see page 742)

WriteTime (see page 748)

21.1.2.3.24 TCustomIniFile.WriteTime Method

Writes a time value to an ini file.

Pascal

```
procedure WriteTime(const Section: string; const Name: string; Value: TDateTime); virtual;
```

C++

```
virtual __fastcall WriteTime(const UnicodeString Section, const UnicodeString Name, TDateTime
Value);
```

Description

Call WriteTime to write a time value to an ini file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the time value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteTime creates the section and key and sets its initial value to Value.

See Also

WriteBool (see page 737)

WriteDate (see page 741)

WriteDateTime (see page 741)

WriteFloat (see page 742)

WriteInteger (see page 742)

WriteString (see page 745)

21.1.2.4 TCustomIniFile Properties

The properties of the TCustomIniFile class are listed here.

Public Properties

	Name	Description
	FileName (see page 749)	Contains the name of the ini file from which to read and to which to write information.

21.1.2.4.1 TCustomIniFile.FileName Property

Contains the name of the ini file from which to read and to which to write information.

Pascal

```
property FileName: string;
```

C++

```
__property UnicodeString FileName;
```

Description

Read FileName to retrieve the name of the ini file that this object can be used to access. The methods that read and write ini file information automatically handle opening and closing the file specified by FileName.

FileName is read-only. The name of the ini file the object uses is supplied as the parameter for the constructor.

See Also

Create (see page 720)

21.1.3 THashedStringList Class

THashedStringList maintains a list of strings using an internal hash table.

Class Hierarchy

Pascal

```
THashedStringList = class(TStringList);
```

C++

```
class THashedStringList : public TStringList;
```

File

IniFiles

Description

THashedStringList is a string list that uses a hash table internally to speed the process of locating strings. It is used internally by TMemIniFile (see page 765) to manage the strings from an INI file, but can be used in the same way as any other string list. By using THashedStringList instead of TStringList, you can improve performance when the list contains a large number of strings.

See Also

TStringList

TMemIniFile (see page 765)

Working with Lists

21.1.3.1 THashedStringList Members

The following tables list the members exposed by THashedStringList.

Protected Methods

	Name	Description
	Changed (see page 751)	This is Changed, a member of class THashedStringList.

Public Methods

	Name	Description
	Destroy (see page 750)	Destroys an instance of THashedStringList (see page 749).
	IndexOf (see page 751)	Returns the position of a string in the list.
	IndexOfName (see page 752)	Returns the position of the first string with the form Name=Value with the specified name part.

21.1.3.2 THashedStringList.Destroy Destructor

Destroys an instance of THashedStringList (see page 749).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~THashedStringList();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the THashedStringList (see page 749) reference is not nil and only then calls Destroy.

See Also

System::TObject::Free

21.1.3.3 THashedStringList Methods

The methods of the THashedStringList class are listed here.

Protected Methods

	Name	Description
	Changed (see page 751)	This is Changed, a member of class THashedStringList.

Public Methods

	Name	Description
	IndexOf (see page 751)	Returns the position of a string in the list.
	IndexOfName (see page 752)	Returns the position of the first string with the form Name=Value with the specified name part.

21.1.3.3.1 THashedStringList.Changed Method

Pascal

```
procedure Changed; override;
```

C++

```
virtual __fastcall Changed();
```

Description

This is Changed, a member of class THashedStringList.

21.1.3.3.2 THashedStringList.IndexOf Method

Returns the position of a string in the list.

Pascal

```
function IndexOf(const S: string): Integer; override;
```

C++

```
virtual __fastcall int IndexOf(const UnicodeString S);
```

Description

Call IndexOf to obtain the position of the first occurrence of a string that matches S. A string matches S if it is identical to S or differs only in case.

IndexOf returns the 0-based index of the string. Thus, if S matches the first string in the list, IndexOf returns 0, if S is the second string, IndexOf returns 1, and so on. If the string does not have a match in the string list, IndexOf returns -1.

See Also

Find

IndexOfName (see page 752)

IndexOfObject

Strings

21.1.3.3.3 THashedStringList.IndexOfName Method

Returns the position of the first string with the form Name=Value with the specified name part.

Pascal

```
function IndexOfName(const Name: string): Integer; override;
```

C++

```
virtual __fastcall int IndexOfName(const UnicodeString Name);
```

Description

Call IndexOfName to locate the first occurrence of a string with the form Name=Value where the name part is equal to the Name parameter or differs only in case. IndexOfName returns the 0-based index of the string. If no string in the list has the indicated name, IndexOfName returns -1.

The Name that identifies the string is to the left of the equal sign (=), and the current Value of the Name identifier is on the right side. There should be no spaces present before or after the equal sign.

Note: If there is more than one string with a name portion matching the Name parameter, IndexOfName returns the position of the first such string.

See Also

IndexOf (see page 751)

Names

Values

IndexOfObject

21.1.4 TIniFile Class

TIniFile stores and retrieves application-specific information and settings from INI files.

Class Hierarchy

Pascal

```
TIniFile = class(TCustomIniFile);
```

C++

```
class TIniFile : public TCustomIniFile;
```

File

IniFiles

Description

TIniFile enables handling the storage and retrieval of application-specific information and settings in a standard INI file. An INI file stores information in logical groupings, called "sections." Within each section, actual data values are stored in named keys. Keys take the form:

```
<keyname>=<value>
```

A FileName (see page 749) is passed to the TIniFile constructor and identifies the INI file that the object accesses.

Note: On Windows, a related object, TMemIniFile (see page 765), works the same way as TIniFile, but buffers writes in memory to minimize disk access.

Note: On Linux, TIniFile is the same as TMemIniFile (see page 765).

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
```

```
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
```

```
    Store current form properties to be used in later sessions.
    }
    try
      SettingsFile.WriteInteger (Name, 'Top', Top);
      SettingsFile.WriteInteger (Name, 'Left', Left);
      SettingsFile.WriteInteger (Name, 'Width', Width);
      SettingsFile.WriteInteger (Name, 'Height', Height);
      SettingsFile.WriteString  (Name, 'Caption', Caption);
      SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
      SettingsFile.WriteDateTime(Name, 'LastRun', Now);
    finally
      SettingsFile.Free;
    end;

  end;

  function TForm2.OpenIniFileInstance: TCustomIniFile;
  begin
    {
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    }

    case RadioGroup1.ItemIndex of
      0:
        begin
          { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
          Result := TRegistryIniFile.Create('Software\' + Application.Title);
        end;
      1:
        begin
          { Ini file mode selected }
          Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
      2:
        begin
          { Memory based Ini file mode selected }
          Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
    end;
  end;
```

See Also

TRegistry (see page 3246)

TRegistryIniFile (see page 3299)

21.1.4.1 TIniFile Members

The following tables list the members exposed by TIniFile.

Public Methods

	Name	Description
	Create (see page 720)	Creates a TCustomIniFile (see page 716) object for an application.
	DeleteKey (see page 757)	Removes an INI file entry for a specified key value.
	Destroy (see page 756)	Disposes of an object instance.
	EraseSection (see page 757)	Erases an entire section of an INI file.
	ReadBinaryStream (see page 722)	Retrieves a binary value from the ini file.
	ReadBool (see page 723)	Retrieves a Boolean value from an ini file.
	ReadDate (see page 726)	Retrieves a date value from an ini file.
	ReadDateTime (see page 726)	Retrieves a date-time value from an ini file.

	ReadFloat (see page 727)	Retrieves a float value from an ini file.
	ReadInteger (see page 727)	Retrieves an integer value from an ini file.
	ReadSection (see page 758)	Reads all the key names from a specified section of an INI file into a string list.
	ReadSections (see page 758)	Reads the names of all sections in an INI file into a string list.
	ReadSections (see page 731)	This is the overview for the ReadSections method overload.
	ReadSectionValues (see page 759)	Reads the values from all keys within a section of an INI file into a string list.
	ReadString (see page 759)	Retrieves a string value from an INI file.
	ReadTime (see page 735)	Retrieves a time value from an ini file.
	SectionExists (see page 736)	Indicates whether a section exists in the ini file.
	UpdateFile (see page 762)	Flushes buffered INI file data to disk.
	ValueExists (see page 736)	Indicates whether a key exists in the ini file.
	WriteBinaryStream (see page 737)	Writes a binary value to the ini file.
	WriteBool (see page 737)	Writes a Boolean value to an ini file.
	WriteDate (see page 741)	Writes a date value to an ini file.
	WriteDateTime (see page 741)	Writes a date-time value to an ini file.
	WriteFloat (see page 742)	Writes a float value to an ini file.
	WriteInteger (see page 742)	Writes an integer value to an ini file.
	WriteString (see page 762)	Writes a string value to an INI file.
	WriteTime (see page 748)	Writes a time value to an ini file.

Public Properties

	Name	Description
	FileName (see page 749)	Contains the name of the ini file from which to read and to which to write information.

21.1.4.2 TIniFile.Destroy Destructor

Disposes of an object instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TIniFile();
```

Description

Do not call System::TObject::Destroy directly. Call System::TObject::Free instead. Free verifies that the object reference is not nil before calling System::TObject::Destroy.

The System::TObject::Destroy method defined by System::TObject deallocates memory. Descendant objects usually define a destructor that is customized for that particular kind of object.

When declaring a System::TObject::Destroy method in a descendant, always add the override directive to the declaration and call the inherited System::TObject::Destroy as the last statement in the overriding method. Since System::TObject::Destroy is a virtual method, overriding it ensures that the proper inherited behavior occurs.

Note: If an exception escapes from the constructor, the destructor is called to destroy the partially constructed object instance that failed to initialize completely. Therefore, destructors should check that allocated resources, such as handles, were actually allocated before trying to release them, since their value might be zero.

System::TObject::Destroy should be implemented so that it calls System::TObject::Free on all subobjects created within the object's constructor (that is, allocated by the constructor). Unlike System::TObject::Destroy, System::TObject::Free provides a safeguard when destroying objects that are nil.

See Also

System::TObject::Free

21.1.4.3 TIniFile Methods

The methods of the TIniFile class are listed here.

Public Methods

	Name	Description
	DeleteKey (see page 757)	Removes an INI file entry for a specified key value.
	EraseSection (see page 757)	Erases an entire section of an INI file.
	ReadSection (see page 758)	Reads all the key names from a specified section of an INI file into a string list.
	ReadSections (see page 758)	Reads the names of all sections in an INI file into a string list.
	ReadSectionValues (see page 759)	Reads the values from all keys within a section of an INI file into a string list.
	ReadString (see page 759)	Retrieves a string value from an INI file.
	UpdateFile (see page 762)	Flushes buffered INI file data to disk.
	WriteString (see page 762)	Writes a string value to an INI file.

21.1.4.3.1 TIniFile.DeleteKey Method

Removes an INI file entry for a specified key value.

Pascal

```
procedure DeleteKey(const Section: String; const Ident: String); override;
```

C++

```
virtual __fastcall DeleteKey(const UnicodeString Section, const UnicodeString Ident);
```

Description

Call DeleteKey to erase an INI file entry. Section is string containing the name of an INI file section, and Ident is a string containing the name of the key from for which to set a nil (Delphi) or NULL (C++) value.

This version of DeleteKey is only available if the application is compiled for Windows. On Linux, DeleteKey is inherited from TMemIniFile (see page 765).

Note: Attempting to delete a key in a nonexistent section or attempting to delete a nonexistent key are not errors. In these cases, DeleteKey does nothing.

See Also

EraseSection (see page 757)

21.1.4.3.2 TIniFile.EraseSection Method

Erases an entire section of an INI file.

Pascal

```
procedure EraseSection(const Section: string); override;
```

C++

```
virtual __fastcall EraseSection(const UnicodeString Section);
```

Description

Call EraseSection to remove a section, all its keys, and their data values from an INI file. Section identifies the INI file section to

remove. If a section cannot be removed, an exception is raised.

This version of EraseSection is only available if the application is compiled for Windows. On Linux, EraseSection is inherited from TMemIniFile (see page 765).

See Also

DeleteKey (see page 757)

21.1.4.3.3 TIniFile.ReadSection Method

Reads all the key names from a specified section of an INI file into a string list.

Pascal

```
procedure ReadSection(const Section: string; Strings: TStrings); override;
```

C++

```
virtual __fastcall ReadSection(const UnicodeString Section, TStrings Strings);
```

Description

Call ReadSection to retrieve the names of all keys within a specified section of an INI file into a string list object.

Section identifies the section from which to retrieve a list of key names. Strings specifies the TStrings descendant to hold the retrieved names. Strings can point to a TStrings descendant such as a string list (TStringList), or to a component property that is of type TStrings, such as the Items property of TListBox.

This version of ReadSection is only available if the application is compiled for Windows. On Linux, ReadSection is inherited from TMemIniFile (see page 765).

See Also

ReadSections (see page 758)

ReadSectionValues (see page 759)

21.1.4.3.4 TIniFile.ReadSections Method

Reads the names of all sections in an INI file into a string list.

Pascal

```
procedure ReadSections(Strings: TStrings); override;
```

C++

```
virtual __fastcall ReadSections(TStrings Strings);
```

Description

Call ReadSections to retrieve the names of all sections in an INI file into a string list object.

Strings specifies the string object to hold the retrieved names. Strings can point to a TStrings object, or to a component property, such as Items for a TListBox component.

This version of ReadSections is only available if the application is compiled for Windows. On Linux, ReadSections is inherited from TMemIniFile (see page 765).

See Also

ReadSection (see page 758)

ReadSectionValues (see page 759)

21.1.4.3.5 TIniFile.ReadSectionValues Method

Reads the values from all keys within a section of an INI file into a string list.

Pascal

```
procedure ReadSectionValues(const Section: string; Strings: TStrings); override;
```

C++

```
virtual __fastcall ReadSectionValues(const UnicodeString Section, TStrings Strings);
```

Description

Call ReadSectionValues to read the keys, and the values from all keys, within a specified section of an INI file into a string list object (TStrings descendant). The key names and values are added to the string list as strings of the form KeyName=ValueName.

Section identifies the section in the file from which to read key values.

Strings is the string list object into which to read the values. Strings can point to a TStrings object, or to a component property, such as Items for a TListBox component, that is a TStrings object itself.

This version of ReadSectionValues is only available if the application is compiled for Windows. On Linux, ReadSectionValues is inherited from TMemIniFile (see page 765).

See Also

ReadSection (see page 758)

ReadSections (see page 758)

21.1.4.3.6 TIniFile.ReadString Method

Retrieves a string value from an INI file.

Pascal

```
function ReadString(const Section: string; const Ident: string; const Default: string):
string; override;
```

C++

```
virtual __fastcall UnicodeString ReadString(const UnicodeString Section, const UnicodeString
Ident, const UnicodeString Default);
```

Description

Call ReadString to read a string value from an INI file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the value. Default is the string value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

This version of ReadString is only available if the application is compiled for Windows. On Linux, ReadString is inherited from TMemIniFile (see page 765).

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
```

```
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
```

```
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
    SettingsFile.WriteInteger (Name, 'Height', Height);
    SettingsFile.WriteString  (Name, 'Caption', Caption);
    SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
    SettingsFile.WriteDateTime(Name, 'LastRun', Now);
  finally
    SettingsFile.Free;
  end;
```

```
end;

function TForm2.OpenIniFileInstance: TCustomIniFile;
begin
  {
  Open/create a new INI file that has the same name as our executable
  only with the INI extension.
  }

  case RadioGroup1.ItemIndex of
    0:
      begin
        { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
        Result := TRegistryIniFile.Create('Software\' + Application.Title);
      end;
    1:
      begin
        { Ini file mode selected }
        Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
    2:
      begin
        { Memory based Ini file mode selected }
        Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
  end;
end;
```

See Also

ReadBool (see page 723)

ReadInteger (see page 727)

21.1.4.3.7 TIniFile.UpdateFile Method

Flushes buffered INI file data to disk.

Pascal

```
procedure UpdateFile; override;
```

C++

```
virtual __fastcall UpdateFile();
```

Description

Call UpdateFile to flush buffered reads from and writes to the INI file to disk. UpdateFile is useful under Windows9x, but has no effect under Windows NT because NT does not buffer INI file reads and writes.

This version of UpdateFile is only available if the application is compiled for Windows. On Linux, Update is inherited from TMemIniFile (see page 765).

See Also

FileName (see page 749)

TMemIniFile (see page 765)

21.1.4.3.8 TIniFile.WriteString Method

Writes a string value to an INI file.

Pascal

```
procedure WriteString(const Section: String; const Ident: String; const Value: String);
override;
```

C++

```
virtual __fastcall WriteString(const UnicodeString Section, const UnicodeString Ident, const
UnicodeString Value);
```

Description

Call WriteString to write a string value to an INI file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the string value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteString creates the section and key and sets its initial value to Value.

This version of WriteString is only available if the application is compiled for Windows. On Linux, WriteString is inherited from TMemIniFile (see page 765).

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);
```

```
        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
```

```
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
    SettingsFile.WriteInteger (Name, 'Height', Height);
    SettingsFile.WriteString  (Name, 'Caption', Caption);
    SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
    SettingsFile.WriteDateTime(Name, 'LastRun', Now);
  finally
    SettingsFile.Free;
  end;

end;

function TForm2.OpenIniFileInstance: TCustomIniFile;
begin
  {
  Open/create a new INI file that has the same name as our executable
  only with the INI extension.
  }

  case RadioGroup1.ItemIndex of
    0:
      begin
        { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
        Result := TRegistryIniFile.Create('Software\' + Application.Title);
      end;
    1:
      begin
        { Ini file mode selected }
        Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
    2:
      begin
        { Memory based Ini file mode selected }
        Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
  end;
end;
```

See Also

WriteBool (see page 737)

WriteInteger (see page 742)

21.1.5 TMemIniFile Class

TMemIniFile enables buffered storage and retrieval of application-specific information and settings in an INI file.

Class Hierarchy

Pascal

```
TMemIniFile = class(TCustomIniFile);
```

C++

```
class TMemIniFile : public TCustomIniFile;
```

File

IniFiles

Description

Use TMemIniFile to store and retrieve application-specific information and settings in a Windows INI file. An INI file stores information in logical groupings, called "sections." Within each section, actual data values are stored in named keys. Keys take the form:

```
<keyname>=<value>
```

TMemIniFile buffers all changes to the INI file. The INI file is read once, when the object is first created. Data from the INI file is stored in nested string lists. Each section in the INI file occupies one element in the top-most string list, and each element in this may itself contain a string list. Each element in each of the contained string list represents a key within the section. After the data is read, any changes to the data are stored in memory. To write the data from memory back to the associated INI file, call the UpdateFile (see page 778) method.

All TMemIniFile methods to read, write, and erase sections, keys, and values operate on the in-memory copy of the INI file.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();
```

```
    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
```

```
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
    SettingsFile.WriteInteger (Name, 'Height', Height);
    SettingsFile.WriteString  (Name, 'Caption', Caption);
    SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
    SettingsFile.WriteDateTime(Name, 'LastRun', Now);
  finally
    SettingsFile.Free;
  end;

end;

function TForm2.OpenIniFileInstance: TCustomIniFile;
begin
  {
  Open/create a new INI file that has the same name as our executable
  only with the INI extension.
  }

  case RadioGroup1.ItemIndex of
    0:
      begin
        { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
        Result := TRegistryIniFile.Create('Software\' + Application.Title);
      end;
    1:
      begin
        { Ini file mode selected }
        Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
    2:
      begin
        { Memory based Ini file mode selected }
        Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
  end;
end;
```

See Also

TStrings

TIniFile (see page 752)

TCustomIniFile (see page 716)

TRegistry (see page 3246)

TRegistryIniFile (see page 3299)

21.1.5.1 TMemIniFile Members

The following tables list the members exposed by TMemIniFile.

Public Methods

	Name	Description
	Clear (see page 771)	Erases all data from the INI file in memory.
	Create (see page 770)	This is the overview for the Create constructor overload.
V	DeleteKey (see page 771)	Deletes a specified entry from the .ini file.
V	Destroy (see page 770)	Frees the memory associated with the TMemIniFile (see page 765) object.
V	EraseSection (see page 772)	Erases an entire section of the INI file.
	GetStrings (see page 772)	Populates a string list with INI file data.
V	ReadBinaryStream (see page 722)	Retrieves a binary value from the ini file.
V	ReadBool (see page 723)	Retrieves a Boolean value from an ini file.
V	ReadDate (see page 726)	Retrieves a date value from an ini file.
V	ReadDateTime (see page 726)	Retrieves a date-time value from an ini file.
V	ReadFloat (see page 727)	Retrieves a float value from an ini file.
V	ReadInteger (see page 727)	Retrieves an integer value from an ini file.
V	ReadSection (see page 773)	Reads all the key names from a specified section of an INI file into a string list.
V	ReadSections (see page 773)	Reads the names of all sections in an INI file into a string list.
A	ReadSections (see page 731)	This is the overview for the ReadSections method overload.
V	ReadSectionValues (see page 774)	Reads the values from all keys within a section of an INI file into a string list.
V	ReadString (see page 774)	Retrieves a string value from the INI file.
V	ReadTime (see page 735)	Retrieves a time value from an ini file.
	Rename (see page 777)	Associates the TMemIniFile (see page 765) with a different INI file.
	SectionExists (see page 736)	Indicates whether a section exists in the ini file.
	SetStrings (see page 778)	Populates the memory buffer with contents of a string list.
V	UpdateFile (see page 778)	Flushes buffered INI file data to disk.
V	ValueExists (see page 736)	Indicates whether a key exists in the ini file.
V	WriteBinaryStream (see page 737)	Writes a binary value to the ini file.
V	WriteBool (see page 737)	Writes a Boolean value to an ini file.
V	WriteDate (see page 741)	Writes a date value to an ini file.
V	WriteDateTime (see page 741)	Writes a date-time value to an ini file.
V	WriteFloat (see page 742)	Writes a float value to an ini file.
V	WriteInteger (see page 742)	Writes an integer value to an ini file.
V	WriteString (see page 779)	Writes a string value to an INI file.
V	WriteTime (see page 748)	Writes a time value to an ini file.

Public Properties

	Name	Description
	CaseSensitive (see page 782)	Controls whether section and key names are handled in a case-sensitive manner.
	Encoding (see page 782)	This is Encoding, a member of class TMemIniFile.

	FileName (see page 749)	Contains the name of the ini file from which to read and to which to write information.

21.1.5.2 Create Constructor

This is the overview for the Create constructor overload.

Overload List

	Name	Description
	TMemIniFile.Create (string) (see page 770)	Creates a TMemIniFile (see page 765) object for an application.
	TMemIniFile.Create (string, TEncoding) (see page 770)	Creates a TMemIniFile (see page 765) object for an application.

21.1.5.2.1 TMemIniFile.Create Constructor (string)

Creates a TMemIniFile (see page 765) object for an application.

Pascal

```
constructor Create(const FileName: string); overload;
constructor Create(const FileName: string; Encoding: TEncoding); overload;
```

C++

```
__fastcall TMemIniFile(const UnicodeString FileName);
__fastcall TMemIniFile(const UnicodeString FileName, TEncoding Encoding);
```

Description

Call Create to instantiate a TMemIniFile (see page 765) object for an application. Create sets the FileName (see page 749) property, and loads a copy of the INI file into memory, in an internal string list.

FileName (see page 749) is the name of the INI file the TMemIniFile (see page 765) uses.

Note: On Windows, INI files are meant to be stored in the \WINDOWS directory or the application's directory. To work with an INI file in another location, specify the full path name of the file in FileName (see page 749).

See Also

FileName (see page 749)

System::TObject::Create

21.1.5.3 TMemIniFile.Destroy Destructor

Frees the memory associated with the TMemIniFile (see page 765) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TMemIniFile();
```

Description

Do not call Destroy directly; instead call Free.

Warning: The destructor does not write the current memory image to disk. Be sure to save changes made to TMemIniFile (see page 765) by calling UpdateFile (see page 778) before destroying a TMemIniFile (see page 765) instance.

See Also

System::TObject::Free

UpdateFile (see page 778)

Create (see page 770)

21.1.5.4 TMemIniFile Methods

The methods of the TMemIniFile class are listed here.

Public Methods

	Name	Description
	Clear (see page 771)	Erases all data from the INI file in memory.
	DeleteKey (see page 771)	Deletes a specified entry from the .ini file.
	EraseSection (see page 772)	Erases an entire section of the INI file.
	GetStrings (see page 772)	Populates a string list with INI file data.
	ReadSection (see page 773)	Reads all the key names from a specified section of an INI file into a string list.
	ReadSections (see page 773)	Reads the names of all sections in an INI file into a string list.
	ReadSectionValues (see page 774)	Reads the values from all keys within a section of an INI file into a string list.
	ReadString (see page 774)	Retrieves a string value from the INI file.
	Rename (see page 777)	Associates the TMemIniFile (see page 765) with a different INI file.
	SetStrings (see page 778)	Populates the memory buffer with contents of a string list.
	UpdateFile (see page 778)	Flushes buffered INI file data to disk.
	WriteString (see page 779)	Writes a string value to an INI file.

21.1.5.4.1 TMemIniFile.Clear Method

Erases all data from the INI file in memory.

Pascal

```
procedure Clear;
```

C++

```
__fastcall Clear();
```

Description

Call Clear to erase all data from the INI file that is currently buffered in memory. All sections, keys, and values are erased.

Warning: No exception is generated when using Clear and the data has not been saved to the INI file with the UpdateFile (see page 778) method.

See Also

UpdateFile (see page 778)

SetStrings (see page 778)

21.1.5.4.2 TMemIniFile.DeleteKey Method

Deletes a specified entry from the .ini file.

Pascal

```
procedure DeleteKey(const Section: String; const Ident: String); override;
```

C++

```
virtual __fastcall DeleteKey(const UnicodeString Section, const UnicodeString Ident);
```

Description

Call DeleteKey to erase .ini file entry. Section is string containing the name of a .ini file section, and Ident is a string containing the name of the key to remove.

Note: Attempting to erase a key in a nonexistent section or attempting to erase a nonexistent key are not errors. In these cases, DeleteKey does nothing.

DeleteKey only affects the in-memory copy of the .ini file, not the copy on disk. The change is not written to disk until you call UpdateFile (see page 778).

See Also

SectionExists (see page 736)

ValueExists (see page 736)

EraseSection (see page 772)

UpdateFile (see page 736)

CaseSensitive (see page 782)

21.1.5.4.3 TMemIniFile.EraseSection Method

Erases an entire section of the INI file.

Pascal

```
procedure EraseSection(const Section: string); override;
```

C++

```
virtual __fastcall EraseSection(const UnicodeString Section);
```

Description

Call EraseSection to remove a section, all its keys, and their data values from an INI file. Section identifies the INI file section to remove. If a section cannot be removed, an exception is raised.

EraseSection only affects the in-memory copy of the INI file, not the copy on disk. The change is not written to disk until you call UpdateFile (see page 778).

See Also

SectionExists (see page 736)

DeleteKey (see page 771)

UpdateFile (see page 778)

CaseSensitive (see page 782)

21.1.5.4.4 TMemIniFile.GetStrings Method

Populates a string list with INI file data.

Pascal

```
procedure GetStrings(List: TStrings);
```

C++

```
__fastcall GetStrings(TStrings List);
```

Description

Call GetStrings to copy the INI file data currently stored in memory into a pre-existing string list.

List is a string list object, created and maintained by the application, into which the INI file data is copied.

See Also

SetStrings (see page 778)

21.1.5.4.5 TMemIniFile.ReadSection Method

Reads all the key names from a specified section of an INI file into a string list.

Pascal

```
procedure ReadSection(const Section: string; Strings: TStrings); override;
```

C++

```
virtual __fastcall ReadSection(const UnicodeString Section, TStrings Strings);
```

Description

Call ReadSection to retrieve the names of all keys within a specified section of an INI file into a string list.

Section identifies the section from which to retrieve a list of key names. Strings specifies the string to hold the retrieved names. Strings can point to a TStrings descendant such as a string list, or to a component property, such as Items for a TListBox component.

Data read by ReadSection comes from the in-memory copy of the INI file, not the copy on disk.

See Also

SectionExists (see page 736)

ReadSections (see page 773)

ReadSectionValues (see page 774)

CaseSensitive (see page 782)

21.1.5.4.6 TMemIniFile.ReadSections Method

Reads the names of all sections in an INI file into a string list.

Pascal

```
procedure ReadSections(Strings: TStrings); override;
```

C++

```
virtual __fastcall ReadSections(TStrings Strings);
```

Description

Call ReadSections to retrieve the names of all sections in an INI file into a string list.

Strings specifies the string object to hold the retrieved names. Strings can point to a TStrings descendant, or to a component property, such as Items for a TListBox component.

Data read by ReadSections comes from the in-memory copy of the INI file, not the copy on disk.

See Also

ReadSection (see page 773)

ReadSectionValues (see page 774)

21.1.5.4.7 TMemIniFile.ReadSectionValues Method

Reads the values from all keys within a section of an INI file into a string list.

Pascal

```
procedure ReadSectionValues(const Section: string; Strings: TStrings); override;
```

C++

```
virtual __fastcall ReadSectionValues(const UnicodeString Section, TStrings Strings);
```

Description

Call ReadSectionValues to read the values from all keys within a specified section of an INI file into a string list object. The values are added to the string list as strings of the form KeyName=ValueName.

Section identifies the section in the file from which to read key values.

Strings is the string list object into which to read the values. Strings can point to a TStrings descendant, or to a component property, such as Items for a TListBox component.

Data read by ReadSectionValues comes from the in-memory copy of the INI file, not the copy on disk.

See Also

SectionExists (see page 736)

ReadSection (see page 773)

ReadSections (see page 773)

CaseSensitive (see page 782)

21.1.5.4.8 TMemIniFile.ReadString Method

Retrieves a string value from the INI file.

Pascal

```
function ReadString(const Section: string; const Ident: string; const Default: string):
string; override;
```

C++

```
virtual __fastcall UnicodeString ReadString(const UnicodeString Section, const UnicodeString
Ident, const UnicodeString Default);
```

Description

Call ReadString to read a string value from an INI file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the value. Default is the string value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

The ReadString method inherited from TCustomIniFile (see page 716) is modified to operate on the in-memory copy of the INI file data. The read methods for all other data types use ReadString internally and then translate to the appropriate data type.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
```

```
      only with the INI extension.
      */

      switch (RadioGroup1->ItemIndex)
      {
          case 0:
              /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
              return new TRegistryIniFile(String("Software\\") + Application->Title);
          case 1:
              /* Ini file mode selected */
              return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
          case 2:
              /* Memory based Ini file mode selected */
              return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
```

```
      SettingsFile.WriteInteger (Name, 'Height', Height);
      SettingsFile.WriteString  (Name, 'Caption', Caption);
      SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
      SettingsFile.WriteDateTime(Name, 'LastRun', Now);
    finally
      SettingsFile.Free;
    end;

  end;

  function TForm2.OpenIniFileInstance: TCustomIniFile;
  begin
    {
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    }

    case RadioGroup1.ItemIndex of
      0:
        begin
          { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
          Result := TRegistryIniFile.Create('Software\' + Application.Title);
        end;
      1:
        begin
          { Ini file mode selected }
          Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
      2:
        begin
          { Memory based Ini file mode selected }
          Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
    end;
  end;
```

See Also

ReadBool (see page 723)

ReadDate (see page 726)

ReadDateTime (see page 726)

ReadFloat (see page 727)

ReadInteger (see page 727)

ReadTime (see page 735)

CaseSensitive (see page 782)

21.1.5.4.9 TMemIniFile.Rename Method

Associates the TMemIniFile (see page 765) with a different INI file.

Pascal

```
procedure Rename(const FileName: string; Reload: Boolean);
```

C++

```
__fastcall Rename(const UnicodeString FileName, Boolean Reload);
```

Description

Call Rename to change the INI file used by the TMemIniFile (see page 765) object to a file other than the one specified when

the object was created.

FileName (see page 749) is name of the new INI file to use.

Reload determines whether data is immediately loaded into memory from the INI file specified in FileName (see page 749). When Reload is true, the TMemIniFile (see page 765) object is repopulated with the data from the specified file. When Reload is false, the in-memory image is unchanged. This is useful, for example, when copying the data from one INI file to another file.

See Also

FileName (see page 749)

Create (see page 770)

21.1.5.4.10 TMemIniFile.SetStrings Method

Populates the memory buffer with contents of a string list.

Pascal

```
procedure SetStrings(List: TStrings);
```

C++

```
__fastcall SetStrings(TStrings List);
```

Description

Call SetStrings to populate the in-memory copy of the INI file data with the contents of a string list.

List is a string list object, created and maintained by the application, from which the INI file data is copied.

SetStrings uses the Clear (see page 771) method to erase any existing INI file data prior to copying the new data.

SetStrings can be used to reread data from the INI file from disk after it is erased with the Clear (see page 771) method. The LoadFromFile method of a string list can be used to retrieve the contents of the INI file and then SetStrings can copy the data into the memory buffer.

See Also

LoadFromFile

Clear (see page 771)

GetStrings (see page 772)

21.1.5.4.11 TMemIniFile.UpdateFile Method

Flushes buffered INI file data to disk.

Pascal

```
procedure UpdateFile; override;
```

C++

```
virtual __fastcall UpdateFile();
```

Description

Call UpdateFile to copy INI file data stored in memory to the copy of the INI file on disk UpdateFile overwrites all data in the disk copy of the INI file with the INI file data stored in memory.

If the Rename (see page 777) method is used to change the INI file the TMemIniFile (see page 765) uses, UpdateFile copies the INI file data in memory to a file of the new name. If the file does not already exist, it is created. If the new file already

exists, it is overwritten.

See Also

FileName (see page 749)

21.1.5.4.12 TMemIniFile.WriteString Method

Writes a string value to an INI file.

Pascal

```
procedure WriteString(const Section: String; const Ident: String; const Value: String);
override;
```

C++

```
virtual __fastcall WriteString(const UnicodeString Section, const UnicodeString Ident, const
UnicodeString Value);
```

Description

Call WriteString to write a string value to an INI file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the string value to write.

The WriteString method inherited from TCustomIniFile (see page 716) is modified to operate on the in-memory copy of the INI file data. The write methods for all other data types use WriteString internally after data values are translated from the original data type.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteString creates the section and key and sets its initial value to Value.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
```

```
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
```

```
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
    SettingsFile.WriteInteger (Name, 'Height', Height);
    SettingsFile.WriteString  (Name, 'Caption', Caption);
    SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
    SettingsFile.WriteDateTime(Name, 'LastRun', Now);
  finally
    SettingsFile.Free;
  end;

end;

function TForm2.OpenIniFileInstance: TCustomIniFile;
begin
  {
  Open/create a new INI file that has the same name as our executable
  only with the INI extension.
  }

  case RadioGroup1.ItemIndex of
    0:
      begin
        { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
        Result := TRegistryIniFile.Create('Software\' + Application.Title);
      end;
    1:
      begin
        { Ini file mode selected }
        Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
    2:
      begin
        { Memory based Ini file mode selected }
        Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
  end;
end;
```

See Also

WriteBool (see page 737)

WriteDate (see page 741)

WriteDateTime (see page 741)

WriteFloat (see page 742)

WriteInteger (see page 742)

WriteTime (see page 748)

CaseSensitive (see page 782)

21.1.5.5 TMemIniFile Properties

The properties of the TMemIniFile class are listed here.

Public Properties

	Name	Description
	CaseSensitive (see page 782)	Controls whether section and key names are handled in a case-sensitive manner.
	Encoding (see page 782)	This is Encoding, a member of class TMemIniFile.

21.1.5.5.1 TMemIniFile.CaseSensitive Property

Controls whether section and key names are handled in a case-sensitive manner.

Pascal

```
property CaseSensitive: Boolean;
```

C++

```
__property Boolean CaseSensitive;
```

Description

Use CaseSensitive to indicate whether the names of sections and keys should be treated in a case-sensitive or case-insensitive manner. Set CaseSensitive to true to allow multiple sections or keys that differ only based on case. Set CaseSensitive to false to treat names that differ only by case as the same.

See Also

ReadSection (see page 773)

ReadString (see page 774)

SectionExists (see page 736)

ValueExists (see page 736)

21.1.5.5.2 TMemIniFile.Encoding Property

Pascal

```
property Encoding: TEncoding;
```

C++

```
__property TEncoding Encoding;
```

Description

This is Encoding, a member of class TMemIniFile.

21.1.6 TStringHash Class

Class Hierarchy

Pascal

```
TStringHash = class;
```

C++

```
class TStringHash;
```

File

IniFiles

Description

This is class IniFiles.TStringHash.

21.1.6.1 TStringHash Members

The following tables list the members exposed by TStringHash.

Protected Methods

	Name	Description
	Find (see page 785)	This is Find, a member of class TStringHash.
V	HashOf (see page 785)	This is HashOf, a member of class TStringHash.

Public Methods

	Name	Description
	Add (see page 784)	This is Add, a member of class TStringHash.
	Clear (see page 784)	This is Clear, a member of class TStringHash.
	Create (see page 783)	This is Create, a member of class TStringHash.
V	Destroy (see page 784)	This is Destroy, a member of class TStringHash.
	Modify (see page 785)	This is Modify, a member of class TStringHash.
	Remove (see page 785)	This is Remove, a member of class TStringHash.
	ValueOf (see page 786)	This is ValueOf, a member of class TStringHash.

21.1.6.2 TStringHash.Create Constructor

Pascal

```
constructor Create(Size: Cardinal = 256);
```

C++

```
__fastcall TStringHash(unsigned Size = 256);
```

Description

This is Create, a member of class TStringHash.

21.1.6.3 TStringHash.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TStringHash();
```

Description

This is Destroy, a member of class TStringHash.

21.1.6.4 TStringHash Methods

The methods of the TStringHash class are listed here.

Protected Methods

	Name	Description
	Find (see page 785)	This is Find, a member of class TStringHash.
	HashOf (see page 785)	This is HashOf, a member of class TStringHash.

Public Methods

	Name	Description
	Add (see page 784)	This is Add, a member of class TStringHash.
	Clear (see page 784)	This is Clear, a member of class TStringHash.
	Modify (see page 785)	This is Modify, a member of class TStringHash.
	Remove (see page 785)	This is Remove, a member of class TStringHash.
	ValueOf (see page 786)	This is ValueOf, a member of class TStringHash.

21.1.6.4.1 TStringHash.Add Method

Pascal

```
procedure Add(const Key: string; Value: Integer);
```

C++

```
__fastcall Add(const UnicodeString Key, int Value);
```

Description

This is Add, a member of class TStringHash.

21.1.6.4.2 TStringHash.Clear Method

Pascal

```
procedure Clear;
```

C++

```
__fastcall Clear();
```

Description

This is Clear, a member of class TStringHash.

21.1.6.4.3 TStringHash.Find Method

Pascal

```
function Find(const Key: string): PPHashItem;
```

C++

```
__fastcall PPHashItem Find(const UnicodeString Key);
```

Description

This is Find, a member of class TStringHash.

21.1.6.4.4 TStringHash.HashOf Method

Pascal

```
function HashOf(const Key: string): Cardinal; virtual;
```

C++

```
virtual __fastcall unsigned HashOf(const UnicodeString Key);
```

Description

This is HashOf, a member of class TStringHash.

21.1.6.4.5 TStringHash.Modify Method

Pascal

```
function Modify(const Key: string; Value: Integer): Boolean;
```

C++

```
__fastcall Boolean Modify(const UnicodeString Key, int Value);
```

Description

This is Modify, a member of class TStringHash.

21.1.6.4.6 TStringHash.Remove Method

Pascal

```
procedure Remove(const Key: string);
```

C++

```
__fastcall Remove(const UnicodeString Key);
```

Description

This is Remove, a member of class TStringHash.

21.1.6.4.7 TStringHash.ValueOf Method

Pascal

```
function ValueOf(const Key: string): Integer;
```

C++

```
__fastcall int ValueOf(const UnicodeString Key);
```

Description

This is ValueOf, a member of class TStringHash.

21.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Records

	Name	Description
	THashItem (see page 787)	This is record IniFiles.THashItem.

21.2.1 IniFiles.THashItem Record

Pascal

```
THashItem = record
  Next: PHashItem;
  Key: string;
  Value: Integer;
end;
```

C++

```
struct THashItem {
  PHashItem Next;
  UnicodeString Key;
  int Value;
};
```

File

IniFiles

Description

This is record IniFiles.THashItem.

21.3 Types

The following table lists types in this documentation.

Types

Name	Description
PHashItem (see page 788)	This is type IniFiles.PHashItem.
PPHashItem (see page 788)	This is type IniFiles.PPHashItem.

21.3.1 IniFiles.PHashItem Type

Pascal

```
PHashItem = ^THashItem;
```

C++

```
^THashItem PHashItem;
```

File

IniFiles

Description

This is type IniFiles.PHashItem.

21.3.2 IniFiles.PPHashItem Type

Pascal

```
PPHashItem = ^PHashItem;
```

C++

```
^PHashItem PPHashItem;
```

File

IniFiles

Description

This is type IniFiles.PPHashItem.

22 IntfInfo Namespace

This is namespace IntfInfo.

Topics

Name	Description
Classes (see page 791)	The following table lists classes in this documentation.
Functions (see page 792)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 797)	The following table lists structs, records, enums in this documentation.
Types (see page 801)	The following table lists types in this documentation.
Constants (see page 804)	The following table lists constants in this documentation.

Classes

	Name	Description
	EInterfaceRTTIException (see page 791)	EInterfaceRTTIException is the exception raised when an interface does not have RTTI.

Constants

Name	Description
CallingConventionName (see page 804)	This is constant IntfInfo.CallingConventionName.
TypeInfoNames (see page 804)	This is constant IntfInfo.TypeInfoNames.

Functions

	Name	Description
	GetClsMemberTypeInfo (see page 792)	Returns the runtime type information (RTTI) for a published member of a class.
	GetDynArrayElTypeInfo (see page 792)	This is function IntfInfo.GetDynArrayElTypeInfo.
	GetDynArrayNextInfo (see page 793)	This is function IntfInfo.GetDynArrayNextInfo.
	GetDynArrayNextInfo2 (see page 793)	This is function IntfInfo.GetDynArrayNextInfo2.
	GetIntfMetaData (see page 793)	This is function IntfInfo.GetIntfMetaData.
	GetIntfMetaData (see page 794)	This is function IntfInfo.GetIntfMetaData.
	GetMethNum (see page 794)	This is function IntfInfo.GetMethNum.
	GetPropListFlat (see page 794)	This is function IntfInfo.GetPropListFlat.
	IsStoredPropConst (see page 795)	This is function IntfInfo.IsStoredPropConst.
	OtherTypeName (see page 795)	This is function IntfInfo.OtherTypeName.
	SameTypeInfo (see page 795)	This is function IntfInfo.SameTypeInfo.
	TypeNamesMatch (see page 796)	This is function IntfInfo.TypeNamesMatch.

Structs, Records, Enums

	Name	Description
	TCallConv (see page 797)	This is record IntfInfo.TCallConv.
	TFillMethodArrayOpt (see page 797)	This is record IntfInfo.TFillMethodArrayOpt.
	TIntfMetaData (see page 798)	TIntfMetaData represents the runtime type information (RTTI) from an invokable interface.
	TIntfMethEntry (see page 799)	TIntfMethEntry and TIntfMethEntryArray (see page 802) represent the table of interface methods in a TInfMetaData value.
	TIntfParamEntry (see page 800)	TIntfParamEntry and TIntfParamEntryArray (see page 802) represents the array of parameters in a TInfMethEntry value.

Types

Name	Description
PIntfMetaData (see page 801)	This is type IntfInfo.PIntfMetaData.
PIntfMethEntry (see page 801)	This is type IntfInfo.PIntfMethEntry.
PIntfParamEntry (see page 802)	This is type IntfInfo.PIntfParamEntry.
TIntfMethEntryArray (see page 802)	This is type IntfInfo.TIntfMethEntryArray.
TIntfParamEntryArray (see page 802)	This is type IntfInfo.TIntfParamEntryArray.
TPIntfMethEntryArray (see page 803)	This is type IntfInfo.TPIntfMethEntryArray.

22.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	EInterfaceRTTIException (see page 791)	EInterfaceRTTIException is the exception raised when an interface does not have RTTI.

22.1.1 EInterfaceRTTIException Class

EInterfaceRTTIException is the exception raised when an interface does not have RTTI.

Class Hierarchy

Pascal

```
EInterfaceRTTIException = class(Exception);
```

C++

```
class EInterfaceRTTIException : public Exception;
```

File

IntfInfo

Description

EInterfaceRTTIException is thrown when an application expects an interface to be invokable but discovers that it does not include any runtime type information (RTTI). For example, the invocation registry raises an EInterfaceRTTIException exception when an application tries to register an interface that is not compiled with RTTI. Invoker components raise an EInterfaceRTTIException exception when an incoming request includes a call to an interface that does not include RTTI.

See Also

TInvokableClassRegistry (see page 840)

TSoapPascalInvoker

IInvokable

22.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	GetClsMemberTypeInfo (see page 792)	Returns the runtime type information (RTTI) for a published member of a class.
	GetDynArrayElTypeInfo (see page 792)	This is function IntfInfo.GetDynArrayElTypeInfo.
	GetDynArrayNextInfo (see page 793)	This is function IntfInfo.GetDynArrayNextInfo.
	GetDynArrayNextInfo2 (see page 793)	This is function IntfInfo.GetDynArrayNextInfo2.
	GetIntfMetaData (see page 793)	This is function IntfInfo.GetIntfMetaData.
	GetIntfMetaData (see page 794)	This is function IntfInfo.GetIntfMetaData.
	GetMethNum (see page 794)	This is function IntfInfo.GetMethNum.
	GetPropListFlat (see page 794)	This is function IntfInfo.GetPropListFlat.
	IsStoredPropConst (see page 795)	This is function IntfInfo.IsStoredPropConst.
	OtherTypeName (see page 795)	This is function IntfInfo.OtherTypeName.
	SameTypeInfo (see page 795)	This is function IntfInfo.SameTypeInfo.
	TypeNamesMatch (see page 796)	This is function IntfInfo.TypeNamesMatch.

22.2.1 IntfInfo.GetClsMemberTypeInfo Function

Returns the runtime type information (RTTI) for a published member of a class.

Pascal

```
function GetClsMemberTypeInfo(const ObjectTypeInfo: PTypeInfo; const MemberName: string = ''):
PTypeInfo;
```

C++

```
PTypeInfo GetClsMemberTypeInfo(const PTypeInfo ObjectTypeInfo, const UnicodeString MemberName
= '');
```

File

IntfInfo

Description

ObjectTypeInfo is a pointer to the type information of the class that includes the published property. You can obtain this value for a class using the TypeInfo (Delphi) or __classid (C++) function.

MemberName is the name of a published property in the class identified by ObjectTypeInfo.

GetClsMemberTypeInfo returns a pointer to the type information for the specified property.

22.2.2 IntfInfo.GetDynArrayElTypeInfo Function

Pascal

```
procedure GetDynArrayElTypeInfo(typeInfo: PTypeInfo; var EltInfo: PTypeInfo; var Dims:
```

```
  Integer);
```

C++

```
  GetDynArrayElTypeInfo(PTypeInfo typeInfo, PTypeInfo EltInfo, int Dims);
```

File

IntfInfo

Description

This is function IntfInfo.GetDynArrayElTypeInfo.

22.2.3 IntfInfo.GetDynArrayNextInfo Function

Pascal

```
  function GetDynArrayNextInfo(typeInfo: PTypeInfo): PTypeInfo;
```

C++

```
  PTypeInfo GetDynArrayNextInfo(PTypeInfo typeInfo);
```

File

IntfInfo

Description

This is function IntfInfo.GetDynArrayNextInfo.

22.2.4 IntfInfo.GetDynArrayNextInfo2 Function

Pascal

```
  function GetDynArrayNextInfo2(typeInfo: PTypeInfo; var Name: string): PTypeInfo;
```

C++

```
  PTypeInfo GetDynArrayNextInfo2(PTypeInfo typeInfo, UnicodeString Name);
```

File

IntfInfo

Description

This is function IntfInfo.GetDynArrayNextInfo2.

22.2.5 IntfInfo.GetIntfMetaData Function

Pascal

```
  procedure GetIntfMetaData(Info: PTypeInfo; var IntfMD: TIntfMetaData; IncludeAllAncMethods:
  Boolean = False); overload;
```

C++

```
  GetIntfMetaData(PTypeInfo Info, TIntfMetaData IntfMD, Boolean IncludeAllAncMethods = False);
```

File

IntfInfo

Description

This is function IntfInfo.GetIntfMetaData.

22.2.6 IntfInfo.GetIntfMetaData Function

Pascal

```
procedure GetIntfMetaData(Info: PTypeInfo; var IntfMD: TIntfMetaData; MethodArrayOpt:
TFillMethodArrayOpt); overload;
```

C++

```
GetIntfMetaData(PTypeInfo Info, TIntfMetaData IntfMD, TFillMethodArrayOpt MethodArrayOpt);
```

File

IntfInfo

Description

This is function IntfInfo.GetIntfMetaData.

22.2.7 IntfInfo.GetMethNum Function

Pascal

```
function GetMethNum(const IntfMD: TIntfMetaData; const MethName: string; ParamCount: Integer =
-1): Integer;
```

C++

```
int GetMethNum(const TIntfMetaData IntfMD, const UnicodeString MethName, int ParamCount = -1);
```

File

IntfInfo

Description

This is function IntfInfo.GetMethNum.

22.2.8 IntfInfo.GetPropListFlat Function

Pascal

```
function GetPropListFlat(TypeInfo: PTypeInfo; out PropList: PPropList): Integer;
```

C++

```
int GetPropListFlat(PTypeInfo TypeInfo, PPropList PropList);
```

File

IntfInfo

Description

This is function IntfInfo.GetPropListFlat.

22.2.9 IntfInfo.IsStoredPropConst Function

Pascal

```
function IsStoredPropConst(Instance: TObject; PropInfo: PPropInfo): Boolean;
```

C++

```
Boolean IsStoredPropConst(TObject * Instance, PPropInfo PropInfo);
```

File

IntfInfo

Description

This is function IntfInfo.IsStoredPropConst.

22.2.10 IntfInfo.OtherTypeName Function

Pascal

```
function OtherTypeName(const TypeName: string): string;
```

C++

```
UnicodeString OtherTypeName(const UnicodeString TypeName);
```

File

IntfInfo

Description

This is function IntfInfo.OtherTypeName.

22.2.11 IntfInfo.SameTypeInfo Function

Pascal

```
function SameTypeInfo(const RegInfo: PTypeInfo; const OtherInfo: PTypeInfo): boolean;
```

C++

```
boolean SameTypeInfo(const PTypeInfo RegInfo, const PTypeInfo OtherInfo);
```

File

IntfInfo

Description

This is function IntfInfo.SameTypeInfo.

22.2.12 IntfInfo.TypeNamesMatch Function

Pascal

```
function TypeNamesMatch(const RegName: string; const OtherName: string): boolean;
```

C++

```
boolean TypeNamesMatch(const UnicodeString RegName, const UnicodeString OtherName);
```

File

IntfInfo

Description

This is function IntfInfo.TypeNamesMatch.

22.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TCallConv (see page 797)	This is record IntfInfo.TCallConv.
	TFillMethodArrayOpt (see page 797)	This is record IntfInfo.TFillMethodArrayOpt.

Records

	Name	Description
	TIntfMetaData (see page 798)	TIntfMetaData represents the runtime type information (RTTI) from an invokable interface.
	TIntfMethEntry (see page 799)	TIntfMethEntry and TIntfMethEntryArray (see page 802) represent the table of interface methods in a TInfMetaData value.
	TIntfParamEntry (see page 800)	TIntfParamEntry and TIntfParamEntryArray (see page 802) represents the array of parameters in a TInfMethEntry value.

22.3.1 IntfInfo.TCallConv Enumeration

Pascal

```
TCallConv = (
  ccReg,
  ccCdecl,
  ccPascal,
  ccStdCall,
  ccSafeCall
);
```

C++

```
enum TCallConv {
  ccReg,
  ccCdecl,
  ccPascal,
  ccStdCall,
  ccSafeCall
};
```

File

IntfInfo

Description

This is record IntfInfo.TCallConv.

22.3.2 IntfInfo.TFillMethodArrayOpt Enumeration

Pascal

```
TFillMethodArrayOpt = (
```

```
    fmoAllBaseMethods,
    fmoRTTIBaseMethods,
    fmoNoBaseMethods
  );
```

C++

```
  enum TFillMethodArrayOpt {
    fmoAllBaseMethods,
    fmoRTTIBaseMethods,
    fmoNoBaseMethods
  };
```

File

IntfInfo

Description

This is record IntfInfo.TFillMethodArrayOpt.

22.3.3 IntfInfo.TIntfMetaData Record

TIntfMetaData represents the runtime type information (RTTI) from an invokable interface.

Pascal

```
  TIntfMetaData = record
    Name: string;
    UnitName: string;
    MDA: TIntfMethEntryArray;
    IID: TGUID;
    Info: PTypeInfo;
    AncInfo: PTypeInfo;
    NumAnc: Integer;
  end;
```

C++

```
  struct TIntfMetaData {
    UnicodeString Name;
    UnicodeString UnitName;
    TIntfMethEntryArray MDA;
    TGUID IID;
    PTypeInfo Info;
    PTypeInfo AncInfo;
    int NumAnc;
  };
```

File

IntfInfo

Description

TIntfMetaData is used by classes that work with the runtime type information (RTTI) extracted from an invokable interface.

Name is the name of the invokable interface.

UnitName is the name of the unit in which the interface is declared.

MDA is an array of records, each of which describes a method on the interface, including its calling conventions and function signature.

IID is the globally unique identifier (GUID) of the interface.

Info is a pointer to the type description for the interface.

AncInfo is a pointer to the type description for the interface's ancestor type.

NumAnc indicates the number of methods that the interface inherits from its ancestor. It provides the offset in MDA of the first method that is introduced by the interface rather than inherited from an ancestor interface.

22.3.4 IntfInfo.TIntfMethEntry Record

TIntfMethEntry and TIntfMethEntryArray (see page 802) represent the table of interface methods in a TInfMetaData value.

Pascal

```
TIntfMethEntry = record
  Name: string;
  CC: TCallConv;
  Pos: Integer;
  ParamCount: Integer;
  ResultInfo: PTypeInfo;
  SelfInfo: PTypeInfo;
  Params: TIntfParamEntryArray;
  HasRTTI: Boolean;
end;
```

C++

```
struct TIntfMethEntry {
  UnicodeString Name;
  TCallConv CC;
  int Pos;
  int ParamCount;
  PTypeInfo ResultInfo;
  PTypeInfo SelfInfo;
  TIntfParamEntryArray Params;
  Boolean HasRTTI;
};
```

File

IntfInfo

Description

TIntfMethEntry describes a single method on an invokable interface.

Name is the name of the method.

CC is the calling convention for that method. It is one of the following values:

Value	Meaning
ccReg	register
ccCdecl	cdecl
ccPascal	pascal
ccStdCall	stdcall
ccSafeCall	safecall

Pos is the position of the method in the interface's method table.

ParamCount is the number of parameters that the method has.

ResultInfo is a pointer to the type description for the return type.

SelfInfo is a pointer to the type description for the method type.

Params is a dynamic array of records describing the method parameters.

HasRTTI indicates whether there is runtime type information (RTTI) for the method. If the method is inherited from an interface that was not compiled with RTTI, HasRTTI is false.

See Also

TIntfMetaData (see page 798)

22.3.5 IntfInfo.TIntfParamEntry Record

TIntfParamEntry and TIntfParamEntryArray (see page 802) represents the array of parameters in a TInfMethEntry value.

Pascal

```
TIntfParamEntry = record
  Flags: TParamFlags;
  Name: string;
  Info: PTypeInfo;
end;
```

C++

```
struct TIntfParamEntry {
  TParamFlags Flags;
  UnicodeString Name;
  PTypeInfo Info;
};
```

File

IntfInfo

Description

TIntfParamEntry describes a single parameter on a method. TIntfParamEntryArray (see page 802) describes all the parameters on a method.

Flags gives the attributes of the parameter.

Info is a pointer to the type description for the parameter type.

See Also

TIntfMethEntry (see page 799)

22.4 Types

The following table lists types in this documentation.

Types

Name	Description
PIntfMetaData (see page 801)	This is type IntfInfo.PIntfMetaData.
PIntfMethEntry (see page 801)	This is type IntfInfo.PIntfMethEntry.
PIntfParamEntry (see page 802)	This is type IntfInfo.PIntfParamEntry.
TIntfMethEntryArray (see page 802)	This is type IntfInfo.TIntfMethEntryArray.
TIntfParamEntryArray (see page 802)	This is type IntfInfo.TIntfParamEntryArray.
TPIntfMethEntryArray (see page 803)	This is type IntfInfo.TPIntfMethEntryArray.

22.4.1 IntfInfo.PIntfMetaData Type

Pascal

```
PIntfMetaData = ^TIntfMetaData;
```

C++

```
^TIntfMetaData PIntfMetaData;
```

File

IntfInfo

Description

This is type IntfInfo.PIntfMetaData.

22.4.2 IntfInfo.PIntfMethEntry Type

Pascal

```
PIntfMethEntry = ^TIntfMethEntry;
```

C++

```
^TIntfMethEntry PIntfMethEntry;
```

File

IntfInfo

Description

This is type IntfInfo.PIntfMethEntry.

22.4.3 IntfInfo.PIntfParamEntry Type

Pascal

```
PIntfParamEntry = ^TIntfParamEntry;
```

C++

```
^TIntfParamEntry PIntfParamEntry;
```

File

IntfInfo

Description

This is type IntfInfo.PIntfParamEntry.

22.4.4 IntfInfo.TIntfMethEntryArray Type

Pascal

```
TIntfMethEntryArray = array of TIntfMethEntry;
```

C++

```
array of TIntfMethEntry TIntfMethEntryArray;
```

File

IntfInfo

Description

This is type IntfInfo.TIntfMethEntryArray.

22.4.5 IntfInfo.TIntfParamEntryArray Type

Pascal

```
TIntfParamEntryArray = array of TIntfParamEntry;
```

C++

```
array of TIntfParamEntry TIntfParamEntryArray;
```

File

IntfInfo

Description

This is type IntfInfo.TIntfParamEntryArray.

22.4.6 IntfInfo.TPIntfMethEntryArray Type

Pascal

```
TPIntfMethEntryArray = array of PIntfMethEntry;
```

C++

```
array of PIntfMethEntry TPIntfMethEntryArray;
```

File

IntfInfo

Description

This is type IntfInfo.TPIntfMethEntryArray.

22.5 Constants

The following table lists constants in this documentation.

Constants

Name	Description
CallingConventionName (see page 804)	This is constant IntfInfo.CallingConventionName.
TypeInfoNames (see page 804)	This is constant IntfInfo.TypeInfoNames.

22.5.1 IntfInfo.CallingConventionName Constant

Pascal

```
CallingConventionName: array[ccReg..ccSafeCall] of string = ('REGISTER', 'CDECL', 'PASCAL',
'STDCALL', 'SAFECALL');
```

C++

```
array[ccReg..ccSafeCall] of string CallingConventionName = ('REGISTER', 'CDECL', 'PASCAL',
'STDCALL', 'SAFECALL');
```

File

IntfInfo

Description

This is constant IntfInfo.CallingConventionName.

22.5.2 IntfInfo.TypeInfoNames Constant

Pascal

```
TypeInfoNames: array[0..37] of string = ( 'Boolean', 'bool', 'AnsiChar', 'char', 'AnsiChar',
'signed char', 'Char', 'wchar_t', 'Byte', 'unsigned char', 'SmallInt', 'short', 'Word',
'unsigned short', 'Integer', 'int', 'Cardinal', 'unsigned int', 'Integer', 'long', 'Cardinal',
'unsigned long', 'Int64', '__int64', 'Int64', 'unsigned __int64', 'Single', 'float', 'Double',
'double', 'Extended', 'long double', 'string', 'UnicodeString', 'AnsiString', 'AnsiString',
'WideString', 'WideString');
```

C++

```
array[0..37] of string TypeInfoNames = ( 'Boolean', 'bool', 'AnsiChar', 'char', 'AnsiChar',
'signed char', 'Char', 'wchar_t', 'Byte', 'unsigned char', 'SmallInt', 'short', 'Word',
'unsigned short', 'Integer', 'int', 'Cardinal', 'unsigned int', 'Integer', 'long', 'Cardinal',
'unsigned long', 'Int64', '__int64', 'Int64', 'unsigned __int64', 'Single', 'float', 'Double',
'double', 'Extended', 'long double', 'string', 'UnicodeString', 'AnsiString', 'AnsiString',
'WideString', 'WideString');
```

File

IntfInfo

Description

This is constant IntfInfo.TypeInfoNames.

23 InvokeRegistry Namespace

This is namespace InvokeRegistry.

Topics

Name	Description
Classes (see page 810)	The following table lists classes in this documentation.
Interfaces (see page 919)	The following table lists interfaces in this documentation.
Functions (see page 929)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 934)	The following table lists structs, records, enums in this documentation.
Types (see page 946)	The following table lists types in this documentation.
Variables (see page 955)	The following table lists variables in this documentation.
Constants (see page 956)	The following table lists constants in this documentation.

Classes

	Name	Description
	ERemotableException (see page 810)	ERemotableException is the class for exceptions that can be communicated from a Web Service application to a client application.
	ETypeRegistryException (see page 813)	ETypeRegistryException is the exception raised when the invocation registry or the invokable type registry encounters an invalid type.
	TDataContext (see page 814)	TDataContext stores temporary information about a set of objects, Variants, and other related values.
	THeaderList (see page 823)	THeaderList maintains a list of SOAP headers.
	TInvContext (see page 829)	TInvContext stores information about a method call on an invokeable interface.
	TInvokableClass (see page 833)	TInvokableClass is the base class for the implementation classes of invokable interfaces.
	TInvokableClassRegistry (see page 840)	TInvokableClassRegistry is the type of the invocation registry.
	TRemotable (see page 868)	TRemotable is the base class for classes that can be passed as parameters or return values in a Web Service application.
	TRemotableTypeRegistry (see page 874)	TRemotableTypeRegistry is the type of the remotable type registry.
	TRemotableXS (see page 892)	TRemotableXS is the base class for classes that represent a non-native scalar type on an invokable interface in a Web Service application.
	TSOAPAttachment (see page 895)	TSOAPAttachment represents an attachment that is included with SOAP messages that represent method calls in a Web Service application.
	TSOAPHeader (see page 908)	TSOAPHeader represents a header on the SOAP envelope that contains an encoded message in a Web Service application.
	TSOAPHeaders (see page 912)	TSOAPHeaders is a helper object for implementing the ISOAPHeaders (see page 923) interface.
	TSOAPHeadersBase (see page 916)	TSOAPHeadersBase is the base class for classes that maintain lists of SOAP headers.

Constants

Name	Description
AS_ATTRIBUTE (see page 956)	Identifies a property of a remotable class as one that corresponds to an attribute in the SOAP encoding.
XMLBase64Types (see page 956)	This is constant InvokeRegistry.XMLBase64Types.
XMLSchemaInstNamepspaces (see page 957)	This is constant InvokeRegistry.XMLSchemaInstNamepspaces.
XMLSchemaNamepspaces (see page 957)	This is constant InvokeRegistry.XMLSchemaNamepspaces.

Functions

	Name	Description
	GetRemotableDataContext (see page 929)	Returns the TDataContext (see page 814) object that provides temporary storage during a call to an invokable interface.
	InvRegistry (see page 930)	Provides access to the invocation registry.
	RemClassRegistry (see page 930)	Provides access to the remotable class registry.
	RemTypeRegistry (see page 931)	Provides access to the remotable type registry.
	SetRemotableDataContext (see page 932)	Sets the data context that provides temporary storage during a call to an invokable interface.
	SubstituteStrings (see page 933)	This is function InvokeRegistry.SubstituteStrings.

Interfaces

	Name	Description
	IHeadersSetter (see page 919)	This is class InvokeRegistry.IHeadersSetter.
	IObjConverter (see page 920)	IObjConverter is the interface for converting between a remotable class and its SOAP representation.
	ISOAPHeaders (see page 923)	ISOAPHeaders is the interface for accessing the headers that accompany a message encoded using SOAP.

Structs, Records, Enums

	Name	Description
	eHeaderMethodType (see page 934)	eHeaderMethodType and THeaderMethodTypeArray (see page 949) indicate the types of messages that should include a given header type.
	ExtNameMapItem (see page 935)	This is record InvokeRegistry.ExtNameMapItem.
	IntfExceptionItem (see page 935)	IntfExceptionItem and TExceptionItemArray (see page 948) describe the association between a remotable exception and an interface that can raise it.
	IntfHeaderItem (see page 936)	IntfHeaderItem and THeaderItemArray (see page 948) describe the association between a header and the SOAP messages that include it.
	InvRegClassEntry (see page 937)	This is record InvokeRegistry.InvRegClassEntry.
	InvRegIntfEntry (see page 937)	InvRegIntfEntry stores information about an invokable interface.
	MethParamNameMapItem (see page 939)	This is record InvokeRegistry.MethParamNameMapItem.
	ObjectConvertOptions (see page 939)	ObjectConvertOptions and TObjectConvertOptions (see page 950) are flags that customize how to convert a remotable object to or from its SOAP representation.
	SerializationOptions (see page 940)	SerializationOptions and TSerializationOptions (see page 953) are flags that customize how to convert a remotable object to or from its SOAP representation.
	TDynToClear (see page 941)	This is record InvokeRegistry.TDynToClear.
	TIntfInvokeOption (see page 942)	TIntfInvokeOption and TIntfInvokeOptions (see page 949) represent flags that customize how to handle calls to an invokable interface.
	TObjMultiOptions (see page 943)	TObjMultiOptions indicates how multiple references to the same remotable class should be translated into a SOAP representation.
	TRemHeaderEntry (see page 944)	This is record InvokeRegistry.TRemHeaderEntry.
	TRemRegEntry (see page 944)	TRemRegEntry stores information for the remotable type registry.

Types

Name	Description
ERemotableExceptionClass (see page 946)	This is type InvokeRegistry.ERemotableExceptionClass.
InvString (see page 947)	InvString is the type for strings that are used to represent all or part of an encoded call on an invokable interface.
InvStringDynArray (see page 947)	This is type InvokeRegistry.InvStringDynArray.

PTRemotable (see page 947)	This is type InvokeRegistry.PTRemotable.
TCreateInstanceProc (see page 948)	TCreateInstanceProc is the type of the factory routine that returns an instance of an invokable class.
TExceptionItemArray (see page 948)	This is type InvokeRegistry.TExceptionItemArray.
THeaderItemArray (see page 948)	This is type InvokeRegistry.THeaderItemArray.
THeaderMethodTypeArray (see page 949)	This is type InvokeRegistry.THeaderMethodTypeArray.
TIntfInvokeOptions (see page 949)	This is type InvokeRegistry.TIntfInvokeOptions.
TInvokableClassClass (see page 949)	This is type InvokeRegistry.TInvokableClassClass.
TInvRegIntfEntryArray (see page 950)	This is type InvokeRegistry.TInvRegIntfEntryArray.
TObjectConvertOptions (see page 950)	This is type InvokeRegistry.TObjectConvertOptions.
TPascalRemotableClassRegistry (see page 950)	TPascalRemotableClassRegistry is the type of the remotable class registry.
TPascalRemotableTypeRegistry (see page 951)	TPascalRemotableTypeRegistry is the type of the remotable type registry.
TRemotableClass (see page 951)	This is type InvokeRegistry.TRemotableClass.
TRemotableClassRegistry (see page 951)	This is type InvokeRegistry.TRemotableClassRegistry.
TRemotableXSClass (see page 952)	This is type InvokeRegistry.TRemotableXSClass.
TRemRegEntryArray (see page 952)	This is type InvokeRegistry.TRemRegEntryArray.
TRequiredArray (see page 952)	TRequiredArray is a dynamic array of boolean values.
TSerializationOptions (see page 953)	This is type InvokeRegistry.TSerializationOptions.
TSOAPAttachmentClass (see page 953)	This is type InvokeRegistry.TSOAPAttachmentClass.
TSOAPHeaderClass (see page 953)	This is type InvokeRegistry.TSOAPHeaderClass.

Variables

Name	Description
AppNameSpacePrefix (see page 955)	Supplies a unique prefix to use for automatically generated namespaces given to invokable interfaces and remotable types.

23.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	ERemotableException (see page 810)	ERemotableException is the class for exceptions that can be communicated from a Web Service application to a client application.
	ETypeRegistryException (see page 813)	ETypeRegistryException is the exception raised when the invocation registry or the invokable type registry encounters an invalid type.
	TDataContext (see page 814)	TDataContext stores temporary information about a set of objects, Variants, and other related values.
	THeaderList (see page 823)	THeaderList maintains a list of SOAP headers.
	TInvContext (see page 829)	TInvContext stores information about a method call on an invokeable interface.
	TInvokableClass (see page 833)	TInvokableClass is the base class for the implementation classes of invokable interfaces.
	TInvokableClassRegistry (see page 840)	TInvokableClassRegistry is the type of the invocation registry.
	TRemotable (see page 868)	TRemotable is the base class for classes that can be passed as parameters or return values in a Web Service application.
	TRemotableTypeRegistry (see page 874)	TRemotableTypeRegistry is the type of the remotable type registry.
	TRemotableXS (see page 892)	TRemotableXS is the base class for classes that represent a non-native scalar type on an invokable interface in a Web Service application.
	TSOAPAttachment (see page 895)	TSOAPAttachment represents an attachment that is included with SOAP messages that represent method calls in a Web Service application.
	TSOAPHeader (see page 908)	TSOAPHeader represents a header on the SOAP envelope that contains an encoded message in a Web Service application.
	TSOAPHeaders (see page 912)	TSOAPHeaders is a helper object for implementing the ISOAPHeaders (see page 923) interface.
	TSOAPHeadersBase (see page 916)	TSOAPHeadersBase is the base class for classes that maintain lists of SOAP headers.

23.1.1 ERemotableException Class

ERemotableException is the class for exceptions that can be communicated from a Web Service application to a client application.

Class Hierarchy

Pascal

```
ERemotableException = class(Exception);
```

C++

```
class ERemotableException : public Exception;
```

File

InvokeRegistry

Description

ERemotableException is the exception class for exceptions that a client raises based on a SOAP fault packet that it receives from a Web Service application. The ERemotableException properties represent standard elements of the SOAP fault packet.

Create (see page 811) an ERemotableException descendant to define a custom exception class for problems encountered when executing the methods of an invokable interface in a Web Service application. When a Web Service application raises an ERemotableException descendant in the course of executing a request from a client application, it adds the values of any published properties to the fault packet, in addition to the value of the Message property. Note that these published properties must be remotable types: that is, the value must be a scalar type, or it must be a registered descendant of TRemotable (see page 868).

Before the properties of an ERemotableException descendant can be transmitted in a fault packet, it must be registered with the remotable type registry (RemTypeRegistry (see page 931)). Call the RegisterXSClass method of the remotable type registry, on both the client and the server application, to enable your exception class to be transmitted.

See Also

RemTypeRegistry (see page 931)

TRemotableTypeRegistry (see page 874)

Creating Custom Exception Classes for Web Services

23.1.1.1 ERemotableException Members

The following tables list the members exposed by ERemotableException.

Public Methods

	Name	Description
	Create (see page 811)	This is the overview for the Create constructor overload.

Public Properties

	Name	Description
	FaultActor (see page 812)	Indicates the agent that raised the fault.
	FaultCode (see page 812)	Indicates where the fault was encountered.
	FaultDetail (see page 813)	Provides application-specific details about the exception that occurred.

23.1.1.2 Create Constructor

This is the overview for the Create constructor overload.

Overload List

	Name	Description
	ERemotableException.Create () (see page 811)	Creates and initializes an instance of ERemotableException (see page 810).
	ERemotableException.Create (string, InvString, InvString) (see page 811)	Creates and initializes an instance of ERemotableException (see page 810).

23.1.1.2.1 ERemotableException.Create Constructor ()

Creates and initializes an instance of ERemotableException (see page 810).

Pascal

```
constructor Create; virtual; overload;
constructor Create(const Msg: string; const AFaultCode: InvString = ''; const AFaultActor:
InvString = ''); virtual; overload;
```

C++

```
virtual __fastcall ERemotableException();
virtual __fastcall ERemotableException(const UnicodeString Msg, const InvString AFaultCode =
'', const InvString AFaultActor = '');
```

Description

Create generates an ERemotableException (see page 810) object.

Msg is the error message associated with the exception. It becomes the value of the Message property.

AFaultCode indicates where the fault that gave rise to the exception was encountered. It becomes the value of the FaultCode (see page 812) property.

AFaultActor indicates the agent that gave rise to the fault. It becomes the value of the FaultActor (see page 812) property.

See Also

Message

FaultCode (see page 812)

FaultActor (see page 812)

23.1.1.3 ERemotableException Properties

The properties of the ERemotableException class are listed here.

Public Properties

	Name	Description
	FaultActor (see page 812)	Indicates the agent that raised the fault.
	FaultCode (see page 812)	Indicates where the fault was encountered.
	FaultDetail (see page 813)	Provides application-specific details about the exception that occurred.

23.1.1.3.1 ERemotableException.FaultActor Property

Indicates the agent that raised the fault.

Pascal

```
property FaultActor: InvString;
```

C++

```
__property InvString FaultActor;
```

Description

FaultActor is the URI of the agent that caused the fault. This is used when the SOAP request passes through multiple agents when processed. The FaultActor can be a proxy server, the Web Service application to which the request was addressed, or another agent that the target Web Service application calls on when processing the request.

This property is only set if the Web Service provider specifies a FaultActor in its SOAP fault packet.

See Also

FaultCode (see page 812)

23.1.1.3.2 ERemotableException.FaultCode Property

Indicates where the fault was encountered.

Pascal

```
property FaultCode: InvString;
```

C++

```
__property InvString FaultCode;
```

Description

FaultCode is one of the following strings:

Value	Meaning
Server	The Web Service application raised the exception before it was able to begin processing the content of the SOAP request packet. For example, if the server can't process a request because it is out of memory, it specifies a FaultCode of 'Server'.
Client	The Web Service application raised the exception because of an error in the SOAP request packet that the client provided. For example, if the SOAP request packet specified a nonexistent method or invalid parameter, the server specifies a FaultCode of 'Client'.
VersionMismatch	The Web Service application is a different version than the one that the SOAP request packet specifies.
MustUnderstand	The Web Service application did not understand one of the SOAP headers in the request packet and the request packet set its MustUnderstand header to indicate that the server must handle all of its SOAP headers.

See Also

FaultActor (see page 812)

23.1.1.3.3 ERemotableException.FaultDetail Property

Provides application-specific details about the exception that occurred.

Pascal

```
property FaultDetail: InvString;
```

C++

```
__property InvString FaultDetail;
```

Description

FaultDetail is a fragment of XML that the server generates to provide detailed information about the exception that occurred. FaultDetail provides context information in addition to the fault message that becomes the value of the Message property.

Each server defines the structure and meaning of FaultDetail in an application-specific way. When an ERemotableException (see page 810) descendant includes published properties, their values are added to FaultDetail.

See Also

Message

23.1.2 ETypeRegistryException Class

ETypeRegistryException is the exception raised when the invocation registry or the invokable type registry encounters an invalid type.

Class Hierarchy

Pascal

```
ETypeRegistryException = class(Exception);
```

C++

```
class ETypeRegistryException : public Exception;
```

File

InvokeRegistry

Description

ETypeRegistryException is thrown when

An application tries to register an implementation class with the invocation registry and that class does not implement any interfaces.

An application tries to look up a type in the remotable type registry that is not remotable.

See Also

TInvokableClassRegistry (see page 840)

TRemotableTypeRegistry (see page 874)

23.1.3 TDataContext Class

TDataContext stores temporary information about a set of objects, Variants, and other related values.

Class Hierarchy

Pascal

```
TDataContext = class;
```

C++

```
class TDataContext;
```

File

InvokeRegistry

Description

TDataContext is a repository for information that belongs together. It introduces a number of methods for storing different types of information, including objects, Variants, pointers, and raw data (untyped bytes). When the TDataContext object is freed, it performs all necessary cleanup of the information it stores, freeing the objects, clearing the Variants, and so on.

Do not use TDataContext directly. It is intended as a base class for classes that need a temporary repository that serves a specific purpose. TDataContext does not allocate the internal dynamic arrays that store pointers and raw data: This task is left to descendant classes, such as TInvContext (see page 829).

See Also

TInvContext (see page 829)

23.1.3.1 TDataContext Members

The following tables list the members exposed by TDataContext.

Protected Fields

	Name	Description
	Data (see page 816)	This is Data, a member of class TDataContext.
	DataOffset (see page 816)	This is DataOffset, a member of class TDataContext.
	DataP (see page 817)	This is DataP, a member of class TDataContext.
	DynArrayToClear (see page 817)	This is DynArrayToClear, a member of class TDataContext.
	FObjsToDestroy (see page 817)	This is FObjsToDestroy, a member of class TDataContext.
	StrToClear (see page 817)	This is StrToClear, a member of class TDataContext.
	UStrToClear (see page 817)	This is UStrToClear, a member of class TDataContext.
	VarToClear (see page 818)	This is VarToClear, a member of class TDataContext.
	WStrToClear (see page 818)	This is WStrToClear, a member of class TDataContext.

Public Methods

	Name	Description
	AddDynArrayToClear (see page 818)	Stores a dynamic array that can be cleared when the data context is finished.
	AddObjectToDestroy (see page 819)	Stores an object that can be freed when the data context is finished.
	AddStrToClear (see page 819)	Stores a string that can be cleared when the data context is finished.
	AddUStrToClear (see page 820)	This is AddUStrToClear, a member of class TDataContext.
	AddVariantToClear (see page 820)	Stores the value of a Variant that can be cleared when the data context is finished.
	AddWStrToClear (see page 821)	Stores a wide string that can be cleared when the data context is finished.
	AllocData (see page 821)	Returns a pointer to the data context's cache of raw memory.
	Create (see page 815)	Creates an instance of TDataContext (see page 814).
	Destroy (see page 815)	Destroys an instance of TDataContext (see page 814).
	GetDataPointer (see page 822)	Returns a pointer from the internal cache of pointers.
	RemoveObjectToDestroy (see page 822)	Remove an object from the list of objects to free when the data context is freed.
	SetDataPointer (see page 822)	Stores a pointer in the internal cache of pointers.

23.1.3.2 TDataContext.Create Constructor

Creates an instance of TDataContext (see page 814).

Pascal

```
constructor Create;
```

C++

```
__fastcall TDataContext();
```

Description

Call Create to instantiate a TDataContext (see page 814) object.

23.1.3.3 TDataContext.Destroy Destructor

Destroys an instance of TDataContext (see page 814).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TDataContext();
```

Description

Destroy frees all the temporary data stored in the TDataContext (see page 814) instance.

See Also

delete

23.1.3.4 TDataContext Fields

The fields of the TDataContext class are listed here.

Protected Fields

	Name	Description
	Data (see page 816)	This is Data, a member of class TDataContext.
	DataOffset (see page 816)	This is DataOffset, a member of class TDataContext.
	DataP (see page 817)	This is DataP, a member of class TDataContext.
	DynArrayToClear (see page 817)	This is DynArrayToClear, a member of class TDataContext.
	FObjsToDestroy (see page 817)	This is FObjsToDestroy, a member of class TDataContext.
	StrToClear (see page 817)	This is StrToClear, a member of class TDataContext.
	UStrToClear (see page 817)	This is UStrToClear, a member of class TDataContext.
	VarToClear (see page 818)	This is VarToClear, a member of class TDataContext.
	WStrToClear (see page 818)	This is WStrToClear, a member of class TDataContext.

23.1.3.4.1 TDataContext.Data Field

Pascal

```
Data: array of Byte;
```

C++

```
array of Byte Data;
```

Description

This is Data, a member of class TDataContext.

23.1.3.4.2 TDataContext.DataOffset Field

Pascal

```
DataOffset: Integer;
```

C++

```
int DataOffset;
```

Description

This is DataOffset, a member of class TDataContext.

23.1.3.4.3 TDataContext.DataP Field

Pascal

```
DataP: array of Pointer;
```

C++

```
array of Pointer DataP;
```

Description

This is DataP, a member of class TDataContext.

23.1.3.4.4 TDataContext.DynArrayToClear Field

Pascal

```
DynArrayToClear: array of TDynToClear;
```

C++

```
array of TDynToClear DynArrayToClear;
```

Description

This is DynArrayToClear, a member of class TDataContext.

23.1.3.4.5 TDataContext.FObjsToDestroy Field

Pascal

```
FObjsToDestroy: array of TObject;
```

C++

```
array of TObject FObjsToDestroy;
```

Description

This is FObjsToDestroy, a member of class TDataContext.

23.1.3.4.6 TDataContext.StrToClear Field

Pascal

```
StrToClear: array of Pointer;
```

C++

```
array of Pointer StrToClear;
```

Description

This is StrToClear, a member of class TDataContext.

23.1.3.4.7 TDataContext.UStrToClear Field

Pascal

```
UStrToClear: array of Pointer;
```

C++

```
array of Pointer UStrToClear;
```

Description

This is UStrToClear, a member of class TDataContext.

23.1.3.4.8 TDataContext.VarToClear Field

Pascal

```
VarToClear: array of Pointer;
```

C++

```
array of Pointer VarToClear;
```

Description

This is VarToClear, a member of class TDataContext.

23.1.3.4.9 TDataContext.WStrToClear Field

Pascal

```
WStrToClear: array of Pointer;
```

C++

```
array of Pointer WStrToClear;
```

Description

This is WStrToClear, a member of class TDataContext.

23.1.3.5 TDataContext Methods

The methods of the TDataContext class are listed here.

Public Methods

	Name	Description
	AddDynArrayToClear (see page 818)	Stores a dynamic array that can be cleared when the data context is finished.
	AddObjectToDestroy (see page 819)	Stores an object that can be freed when the data context is finished.
	AddStrToClear (see page 819)	Stores a string that can be cleared when the data context is finished.
	AddUStrToClear (see page 820)	This is AddUStrToClear, a member of class TDataContext.
	AddVariantToClear (see page 820)	Stores the value of a Variant that can be cleared when the data context is finished.
	AddWStrToClear (see page 821)	Stores a wide string that can be cleared when the data context is finished.
	AllocData (see page 821)	Returns a pointer to the data context's cache of raw memory.
	GetDataPointer (see page 822)	Returns a pointer from the internal cache of pointers.
	RemoveObjectToDestroy (see page 822)	Remove an object from the list of objects to free when the data context is freed.
	SetDataPointer (see page 822)	Stores a pointer in the internal cache of pointers.

23.1.3.5.1 TDataContext.AddDynArrayToClear Method

Stores a dynamic array that can be cleared when the data context is finished.

Pascal

```
procedure AddDynArrayToClear(P: Pointer; Info: PTypeInfo);
```

C++

```
__fastcall AddDynArrayToClear(void * P, PTypeInfo Info);
```

Description

Call AddDynArrayToClear to add a dynamic array to the list of temporary dynamic arrays that TDataContext (see page 814) maintains. When the TDataContext (see page 814) instance is freed, it clears all dynamic arrays in this list of temporary arrays.

P is a pointer to the dynamic array.

Info is the TypeInfo pointer for the dynamic array.

Note: TDataContext (see page 814) does not provide access to this list. The caller must keep a reference to the dynamic arrays for use while the TDataContext (see page 814) instance exists.

See Also

Destroy (see page 815)

AddObjectToDestroy (see page 819)

AddVariantToClear (see page 820)

AddStrToClear (see page 819)

AddWStrToClear (see page 821)

23.1.3.5.2 TDataContext.AddObjectToDestroy Method

Stores an object that can be freed when the data context is finished.

Pascal

```
procedure AddObjectToDestroy(Obj: TObject);
```

C++

```
__fastcall AddObjectToDestroy(TObject * Obj);
```

Description

Call AddObjectToDestroy to add an object instance to the list of temporary objects that TDataContext (see page 814) maintains. When the TDataContext (see page 814) instance is freed, it frees all the objects in this list of temporary objects.

Note: TDataContext (see page 814) does not provide access to this list. The caller must keep a reference to the object for use while the TDataContext (see page 814) instance exists.

See Also

Destroy (see page 815)

AddDynArrayToClear (see page 818)

AddVariantToClear (see page 820)

RemoveObjectToDestroy (see page 822)

AddStrToClear (see page 819)

23.1.3.5.3 TDataContext.AddStrToClear Method

Stores a string that can be cleared when the data context is finished.

Pascal

```
procedure AddStrToClear(P: Pointer);
```

C++

```
__fastcall AddStrToClear(void * P);
```

Description

Call AddStrToClear to add a string to the list of temporary strings that TDataContext (see page 814) maintains. When the TDataContext (see page 814) instance is freed, it clears all strings in this list.

P is a pointer to the string.

Note: TDataContext (see page 814) does not provide access to this list. The caller must keep a reference to the string for use while the TDataContext (see page 814) instance exists.

See Also

Destroy (see page 815)

AddObjectToDestroy (see page 819)

AddVariantToClear (see page 820)

AddDynArrayToClear (see page 818)

AddWStrToClear (see page 821)

23.1.3.5.4 TDataContext.AddUStrToClear Method

Pascal

```
procedure AddUStrToClear(P: Pointer);
```

C++

```
__fastcall AddUStrToClear(void * P);
```

Description

This is AddUStrToClear, a member of class TDataContext.

23.1.3.5.5 TDataContext.AddVariantToClear Method

Stores the value of a Variant that can be cleared when the data context is finished.

Pascal

```
procedure AddVariantToClear(P: PVarData);
```

C++

```
__fastcall AddVariantToClear(PVarData P);
```

Description

Call AddVariantToClear to add a Variant to the list of temporary Variant values that TDataContext (see page 814) maintains. When the TDataContext (see page 814) instance is freed, it clears all the Variants in this list of temporary Variant values.

Note: TDataContext (see page 814) does not provide access to this list. The caller must keep a reference to the Variant for use while the TDataContext (see page 814) instance exists.

See Also

Destroy (see page 815)

AddStrToClear (see page 819)

AddWStrToClear (see page 821)

23.1.3.5.6 TDataContext.AddWStrToClear Method

Stores a wide string that can be cleared when the data context is finished.

Pascal

```
procedure AddWStrToClear(P: Pointer);
```

C++

```
__fastcall AddWStrToClear(void * P);
```

Description

Call AddWStrToClear to add a wide string to the list of temporary wide strings that TDataContext (see page 814) maintains. When the TDataContext (see page 814) instance is freed, it clears all strings in this list.

P is a pointer to the WideString.

Note: TDataContext (see page 814) does not provide access to this list. The caller must keep a reference to the string for use while the TDataContext (see page 814) instance exists.

See Also

Destroy (see page 815)

AddObjectToDestroy (see page 819)

AddVariantToClear (see page 820)

AddDynArrayToClear (see page 818)

AddStrToClear (see page 819)

23.1.3.5.7 TDataContext.AllocData Method

Returns a pointer to the data context's cache of raw memory.

Pascal

```
function AllocData(Size: Integer): Pointer;
```

C++

```
__fastcall void * AllocData(int Size);
```

Description

Use AllocData to obtain a pointer to a chunk of temporary memory.

Size is the number of bytes desired.

AllocData returns a pointer to Size bytes.

Note: TDataContext (see page 814) does not actually allocate the memory that AllocData returns. Descendant classes must initialize the internal cache that the returned pointer addresses.

See Also

GetDataPointer (see page 822)

23.1.3.5.8 TDataContext.GetDataPointer Method

Returns a pointer from the internal cache of pointers.

Pascal

```
function GetDataPointer(Index: Integer): Pointer;
```

C++

```
__fastcall void * GetDataPointer(int Index);
```

Description

GetDataPointer returns a pointer from the internal cache of pointers. This value is assigned by a call to the SetDataPointer (see page 822) method.

Index is the index of the desired pointer, where 0 is the first pointer, 1 is the second pointer, and so on.

Note: TDataContext (see page 814) does not actually allocate the memory to store these pointers. Descendant classes must initialize the internal dynamic array that TDataContext (see page 814) uses to store pointers.

See Also

SetDataPointer (see page 822)

23.1.3.5.9 TDataContext.RemoveObjectToDestroy Method

Remove an object from the list of objects to free when the data context is freed.

Pascal

```
procedure RemoveObjectToDestroy(Obj: TObject);
```

C++

```
__fastcall RemoveObjectToDestroy(TObject * Obj);
```

Description

Call RemoveObjectToDestroy to remove an object instance from the list of temporary objects that TDataContext (see page 814) maintains. This prevents TDataContext (see page 814) from freeing the object when it is itself freed.

For example, TRemotable (see page 868) descendants that are created by a Web Service application when it executes an invokable interface call are automatically added to a data context that persists for the length of the method call. When an application changes the remotable object's DataContext property, it automatically calls RemoveObjectToDestroy so that the object is not automatically freed when the method call is complete.

See Also

DataContext (see page 873)

AddObjectToDestroy (see page 819)

23.1.3.5.10 TDataContext.SetDataPointer Method

Stores a pointer in the internal cache of pointers.

Pascal

```
procedure SetDataPointer(Index: Integer; P: Pointer);
```

C++

```
__fastcall SetDataPointer(int Index, void * P);
```

Description

SetDataPointer stores a pointer that can later be retrieved by a call to GetDataPointer (see page 822). TDataContext (see page 814) owns only the pointer, not the data to which it points. When you free the TDataContext (see page 814) instance, it frees its list of pointers, but not the memory to which they point, unless they are first obtained by a call to the AllocData (see page 821) method.

Index is the index of the pointer in the internal cache of pointers, where 0 specifies the first pointer, 1 specifies the second pointer, and so on.

P is the pointer to store.

Note: TDataContext (see page 814) does not actually allocate the memory to store these pointers. Descendant classes must initialize the internal dynamic array that TDataContext (see page 814) uses to store pointers.

See Also

GetDataPointer (see page 822)

AllocData (see page 821)

23.1.4 THeaderList Class

THeaderList maintains a list of SOAP headers.

Class Hierarchy

InvokeRegistry.THeaderList

Pascal

```
THeaderList = class;
```

C++

```
class THeaderList;
```

File

InvokeRegistry

Description

Use THeaderList to store and maintain a list of SOAP headers (TSOAPHeader (see page 908) descendants).

See Also

TSOAPHeader (see page 908)

23.1.4.1 THeaderList Members

The following tables list the members exposed by THeaderList.

Protected Methods

	Name	Description
	GetCount (see page 826)	This is GetCount, a member of class THeaderList.
	GetHeader (see page 827)	This is GetHeader, a member of class THeaderList.
	GetOwnsObjects (see page 827)	This is GetOwnsObjects, a member of class THeaderList.
	SetOwnsObjects (see page 827)	This is SetOwnsObjects, a member of class THeaderList.

Public Methods

	Name	Description
	Add (see page 825)	This is the overview for the Add method overload.
	Clear (see page 826)	Empties the list and frees the TSOAPHeader (see page 908) objects.
	Create (see page 824)	Creates and initializes a THeaderList (see page 823) instance.
V	Destroy (see page 824)	Destroys an instance of THeaderList (see page 823).
	Extract (see page 826)	Removes a specified header from the list without freeing it.
	IndexOf (see page 827)	This is IndexOf, a member of class THeaderList.

Public Properties

	Name	Description
R	Count (see page 828)	Indicates the number of headers in the list.
R	Headers (see page 828)	Lists the headers.
	OwnsObjects (see page 829)	Allows THeaderList (see page 823) to free objects when they the list is cleared or destroyed.

23.1.4.2 THeaderList.Create Constructor

Creates and initializes a THeaderList (see page 823) instance.

Pascal

```
constructor Create;
```

C++

```
__fastcall THeaderList();
```

Description

Create creates a THeaderList (see page 823) instance.

23.1.4.3 THeaderList.Destroy Destructor

Destroys an instance of THeaderList (see page 823).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~THeaderList();
```

Description

Do not call Destroy directly. Instead, call Free. Free verifies that the THeaderList (see page 823) reference is not nil, and only then calls Destroy.

Destroy frees the THeaderList (see page 823) instance. If the OwnsObjects (see page 829) property is true, it also frees the header objects in the list.

See Also

System::TObject::Free

OwnsObjects (see page 829)

23.1.4.4 THeaderList Methods

The methods of the THeaderList class are listed here.

Protected Methods

	Name	Description
	GetCount (see page 826)	This is GetCount, a member of class THeaderList.
	GetHeader (see page 827)	This is GetHeader, a member of class THeaderList.
	GetOwnsObjects (see page 827)	This is GetOwnsObjects, a member of class THeaderList.
	SetOwnsObjects (see page 827)	This is SetOwnsObjects, a member of class THeaderList.

Public Methods

	Name	Description
	Add (see page 825)	This is the overview for the Add method overload.
	Clear (see page 826)	Empties the list and frees the TSOAPHeader (see page 908) objects.
	Extract (see page 826)	Removes a specified header from the list without freeing it.
	IndexOf (see page 827)	This is IndexOf, a member of class THeaderList.

23.1.4.4.1 Add Method

This is the overview for the Add method overload.

Overload List

	Name	Description
	THeaderList.Add (TObject) (see page 825)	Adds a new header to the end of the list.
	THeaderList.Add (TSOAPHeader) (see page 825)	Adds a new header to the end of the list.

23.1.4.4.1.1 THeaderList.Add Method (TObject)

Adds a new header to the end of the list.

Pascal

```
procedure Add(Header: TSOAPHeader); overload;
procedure Add(Header: TObject); overload;
```

C++

```
__fastcall Add(TSOAPHeader Header);
__fastcall Add(TObject * Header);
```

Description

Call Add to add a header to the end of the Headers (see page 828) property array. When you call Add, the new Header is added to the end of the array and the value of the Count (see page 828) property is incremented.

Header is the TSOAPHeader (see page 908) descendants that is added to the list.

See Also

Headers (see page 828)

Count (see page 828)

Clear (see page 826)

23.1.4.4.2 THeaderList.Clear Method

Empties the list and frees the TSOAPHeader (see page 908) objects.

Pascal

```
procedure Clear;
```

C++

```
__fastcall Clear();
```

Description

Clear removes all of the headers from the list and, if the OwnsObjects (see page 829) property is true, frees each one . After a call to Clear, the value of the Count (see page 828) property is 0.

See Also

Headers (see page 828)

Count (see page 828)

OwnsObjects (see page 829)

23.1.4.4.3 THeaderList.Extract Method

Removes a specified header from the list without freeing it.

Pascal

```
function Extract(Obj: TObject): TObject;
```

C++

```
__fastcall TObject * Extract(TObject * Obj);
```

Description

Call Extract to remove a header from the list without freeing the header object. After a header is removed, all the headers that follow it are moved up in index position and Count (see page 828) is decremented.

See Also

Add (see page 825)

Count (see page 828)

Headers (see page 828)

OwnsObjects (see page 829)

23.1.4.4.4 THeaderList.GetCount Method

Pascal

```
function GetCount: Integer;
```

C++

```
__fastcall int GetCount();
```

Description

This is GetCount, a member of class THeaderList.

23.1.4.4.5 THeaderList.GetHeader Method

Pascal

```
function GetHeader(Index: Integer): TObject;
```

C++

```
__fastcall TObject * GetHeader(int Index);
```

Description

This is GetHeader, a member of class THeaderList.

23.1.4.4.6 THeaderList.GetOwnsObjects Method

Pascal

```
function GetOwnsObjects: Boolean;
```

C++

```
__fastcall Boolean GetOwnsObjects();
```

Description

This is GetOwnsObjects, a member of class THeaderList.

23.1.4.4.7 THeaderList.IndexOf Method

Pascal

```
function IndexOf(Obj: TObject): Integer;
```

C++

```
__fastcall int IndexOf(TObject * Obj);
```

Description

This is IndexOf, a member of class THeaderList.

23.1.4.4.8 THeaderList.SetOwnsObjects Method

Pascal

```
procedure SetOwnsObjects(Val: Boolean);
```

C++

```
__fastcall SetOwnsObjects(Boolean Val);
```

Description

This is SetOwnsObjects, a member of class THeaderList.

23.1.4.5 THeaderList Properties

The properties of the THeaderList class are listed here.

Public Properties

	Name	Description
	Count (see page 828)	Indicates the number of headers in the list.
	Headers (see page 828)	Lists the headers.
	OwnsObjects (see page 829)	Allows THeaderList (see page 823) to free objects when they the list is cleared or destroyed.

23.1.4.5.1 THeaderList.Count Property

Indicates the number of headers in the list.

Pascal

```
property Count: Integer;
```

C++

```
__property int Count;
```

Description

Count indicates the number of headers that the Headers (see page 828) property lists. To change the value of Count, you can add headers using the Add (see page 825) method, remove them using the Extract (see page 826) method, or remove all headers using the Clear (see page 826) method.

See Also

Headers (see page 828)

Add (see page 825)

Clear (see page 826)

Extract (see page 826)

23.1.4.5.2 THeaderList.Headers Property

Lists the headers.

Pascal

```
property Headers [Index: Integer]: TObject;
```

C++

```
__property TObject * Headers[int Index];
```

Description

Use Headers to access headers in the list. Headers is a zero-based array of TSOAPHeader (see page 908) descendants: The first header is indexed as 0, the second header is indexed as 1, and so forth. You can read the value at a specific index, or use Headers with the Count (see page 828) property to iterate through the list.

Note: In Delphi, Headers is the default property of THeaderList (see page 823). This means you can omit the property name.

Note: In C++, you can use the subscript operator to achieve a similar effect.

See Also

OwnsObjects (see page 829)

Add (see page 825)

Extract (see page 826)

Count (see page 828)

Clear (see page 826)

23.1.4.5.3 THeaderList.OwnsObjects Property

Allows THeaderList (see page 823) to free objects when they the list is cleared or destroyed.

Pascal

```
property OwnsObjects: Boolean;
```

C++

```
__property Boolean OwnsObjects;
```

Description

OwnsObjects allows THeaderList (see page 823) to control the memory of its TSOAPHeader (see page 908) objects. If OwnsObjects is true,

calling Clear (see page 826) frees all the headers in the list in addition to emptying the list.

freeing the THeaderList (see page 823) object frees all the objects in the list in addition to destroying the THeaderList (see page 823) itself.

Note: Even if OwnsObjects is true, the Extract (see page 826) method removes objects from the list without freeing them.

See Also

Headers (see page 828)

Clear

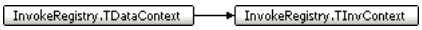
Destroy (see page 824)

Extract (see page 826)

23.1.5 TInvContext Class

TInvContext stores information about a method call on an invokeable interface.

Class Hierarchy



Pascal

```
TInvContext = class(TDataContext);
```

C++

```
class TInvContext : public TDataContext;
```

File

InvokeRegistry

Description

TInvContext is a helper object created and used by TRIO and TSoapPascalInvoker for generating calls to the methods on an invokeable interface. It stores temporary data needed when making the method call. TInvContext introduces methods for

retrieving and setting information about the method being called, including its type signature, parameter values, and so on.

Component writers implementing their own classes to implement the IOPConvert interface must translate between the information stored in a TInvContext object and their class-specific encoding of a method call.

See Also

IOPConvert (see page 2730)

TRIO (see page 4056)

TSoapPascalInvoker

23.1.5.1 TInvContext Members

The following tables list the members exposed by TInvContext.

Protected Fields

	Name	Description
	Data (see page 816)	This is Data, a member of class TDataContext.
	DataOffset (see page 816)	This is DataOffset, a member of class TDataContext.
	DataP (see page 817)	This is DataP, a member of class TDataContext.
	DynArrayToClear (see page 817)	This is DynArrayToClear, a member of class TDataContext.
	FObjsToDestroy (see page 817)	This is FObjsToDestroy, a member of class TDataContext.
	StrToClear (see page 817)	This is StrToClear, a member of class TDataContext.
	UStrToClear (see page 817)	This is UStrToClear, a member of class TDataContext.
	VarToClear (see page 818)	This is VarToClear, a member of class TDataContext.
	WStrToClear (see page 818)	This is WStrToClear, a member of class TDataContext.

Public Methods

	Name	Description
	AddDynArrayToClear (see page 818)	Stores a dynamic array that can be cleared when the data context is finished.
	AddObjectToDestroy (see page 819)	Stores an object that can be freed when the data context is finished.
	AddStrToClear (see page 819)	Stores a string that can be cleared when the data context is finished.
	AddUStrToClear (see page 820)	This is AddUStrToClear, a member of class TDataContext.
	AddVariantToClear (see page 820)	Stores the value of a Variant that can be cleared when the data context is finished.
	AddWStrToClear (see page 821)	Stores a wide string that can be cleared when the data context is finished.
	AllocData (see page 821)	Returns a pointer to the data context's cache of raw memory.
	AllocServerData (see page 831)	Allocates memory for the parameters and result values of a specified method and sets up the parameter and result pointers.
	Create (see page 815)	Creates an instance of TDataContext (see page 814).
	Destroy (see page 815)	Destroys an instance of TDataContext (see page 814).
	GetDataPointer (see page 822)	Returns a pointer from the internal cache of pointers.
	GetParamPointer (see page 831)	Returns a pointer to the value of a parameter on the called method.
	GetResultPointer (see page 832)	Returns a pointer to the return value of the method call.
	RemoveObjectToDestroy (see page 822)	Remove an object from the list of objects to free when the data context is freed.
	SetDataPointer (see page 822)	Stores a pointer in the internal cache of pointers.
	SetMethodInfo (see page 832)	Initializes the TInvContext (see page 829) for a particular method on the invokeable interface.
	SetParamPointer (see page 833)	Sets the value of a specified parameter.
	SetResultPointer (see page 833)	Sets the value of the function's return value.

23.1.5.2 TInvContext Methods

The methods of the TInvContext class are listed here.

Public Methods

	Name	Description
	AllocServerData (see page 831)	Allocates memory for the parameters and result values of a specified method and sets up the parameter and result pointers.
	GetParamPointer (see page 831)	Returns a pointer to the value of a parameter on the called method.
	GetResultPointer (see page 832)	Returns a pointer to the return value of the method call.
	SetMethodInfo (see page 832)	Initializes the TInvContext (see page 829) for a particular method on the invokeable interface.
	SetParamPointer (see page 833)	Sets the value of a specified parameter.
	SetResultPointer (see page 833)	Sets the value of the function's return value.

23.1.5.2.1 TInvContext.AllocServerData Method

Allocates memory for the parameters and result values of a specified method and sets up the parameter and result pointers.

Pascal

```
procedure AllocServerData(const MD: TIntfMethEntry);
```

C++

```
__fastcall AllocServerData(const TIntfMethEntry MD);
```

Description

AllocServerData suballocates the memory that the SetMethodInfo (see page 832) method allocates so that it can represent the parameters and return value of a method. It looks up the types of each parameter and of the return type for the method, allocates memory for their values using the AllocData (see page 821) method, and stores the pointers to that memory by calling the SetParamPointer (see page 833) and SetResultPointer (see page 833) methods.

MD is a description of the method, including its type, calling convention, and parameter signature.

Note: AllocServerData must come after a call to SetMethodInfo (see page 832) so that the pool of memory is available. After calling AllocServerData, an application can call GetParamPointer (see page 831) and GetResultPointer (see page 832) to obtain pointers to the memory allocated for each parameter and for the return value of the method.

See Also

AllocData (see page 821)

SetParamPointer (see page 833)

SetResultPointer (see page 833)

SetMethodInfo (see page 832)

23.1.5.2.2 TInvContext.GetParamPointer Method

Returns a pointer to the value of a parameter on the called method.

Pascal

```
function GetParamPointer(Param: Integer): Pointer;
```

C++

```
__fastcall void * GetParamPointer(int Param);
```

Description

GetParamPointer returns a pointer to the value of a specified parameter.

Param is the index of the desired parameter, where 0 is the first parameter, 1 is the second parameter, and so on.

This value is assigned by a call to the SetParamPointer (see page 833) method.

See Also

SetParamPointer (see page 833)

23.1.5.2.3 TInvContext.GetResultPointer Method

Returns a pointer to the return value of the method call.

Pascal

```
function GetResultPointer: Pointer;
```

C++

```
__fastcall void * GetResultPointer();
```

Description

GetResultPointer returns a pointer to the return value of the method call. This value is assigned by a call to the SetResultPointer (see page 833) method.

See Also

SetResultPointer (see page 833)

23.1.5.2.4 TInvContext.SetMethodInfo Method

Initializes the TInvContext (see page 829) for a particular method on the invokeable interface.

Pascal

```
procedure SetMethodInfo(const MD: TIntfMethEntry);
```

C++

```
__fastcall SetMethodInfo(const TIntfMethEntry MD);
```

Description

SetMethodInfo initializes the invocation context so that it represents a particular method. Objects that implement the IOPConvert interface call this method to initialize an invocation context when implementing their MsgToInvContext method.

MD is a description of the method, including its type, calling convention, and parameter signature.

Note: SetMethodInfo allocates the memory used internally to store pointers and raw data. An application must call this method before any calls to the AllocData (see page 821), SetDataPointer (see page 822), or SetParamPointer (see page 833) methods.

See Also

IOPConvert (see page 2730)

MsgToInvContext (see page 2735)

AllocData (see page 821)

SetDataPointer (see page 822)

SetParamPointer (see page 833)

23.1.5.2.5 TInvContext.SetParamPointer Method

Sets the value of a specified parameter.

Pascal

```
procedure SetParamPointer(Param: Integer; P: Pointer);
```

C++

```
__fastcall SetParamPointer(int Param, void * P);
```

Description

SetParamPointer associates a pointer with a specified parameter.

Param identifies the parameter whose value is to be set, where 0 specifies the first parameter, 1 specifies the second parameter, and so on.

P is a pointer to the value of the parameter. Memory to contain the value of P can be allocated using the AllocData (see page 821) method.

See Also

GetParamPointer (see page 831)

AllocData (see page 821)

23.1.5.2.6 TInvContext.SetResultPointer Method

Sets the value of the function's return value.

Pascal

```
procedure SetResultPointer(P: Pointer);
```

C++

```
__fastcall SetResultPointer(void * P);
```

Description

SetResultPointer associates a pointer with the return value of the function.

P is a pointer to the return value. Memory to contain the value to which P points can be allocated using the AllocData (see page 821) method.

See Also

GetResultPointer (see page 832)

AllocData (see page 821)

23.1.6 TInvokableClass Class

TInvokableClass is the base class for the implementation classes of invokable interfaces.

Class Hierarchy

Pascal

```
TInvokableClass = class(TObject, IInterface);
```

C++

```
class TInvokableClass : public TObject, public IInterface;
```

File

InvokeRegistry

Description

Use TInvokableClass as a base class when writing the implementation class of an invokable interface. TInvokableClass has the following important attributes:

The invocation registry (InvRegistry (see page 930)) knows how to create instances of TInvokableClass and (because it has a virtual constructor) its descendants. This allows the registry to supply an invoker in a Web Service application with an instance of the invokable class that can handle an incoming request.

TInvokableClass is an interfaced object that frees itself when the reference count on its interface drops to zero. Invoker components do not know when to free the implementation classes of the interfaces they call. Because TInvokableClass knows when to free itself, you do not need to supply your own lifetime management for this object.

TInvokableClass supports the ISOAPHeaders (see page 923) interface so that your server can process the headers that accompany requests and add headers to outgoing responses.

To use a TInvokableClass descendant in a Web Service application, you must register it with the invocation registry. Use the RegisterInvokableClass method of the invocation registry to register your TInvokableClass descendant.

Note: If you do not use TInvokableClass as a base class when writing the implementation class for an invokable interface, you must supply the invocation registry with a method to retrieve instances of your class and you must handle the lifetime management of your class. If you are sending or receiving SOAP headers, you must use the global GetSOAPHeaders function to obtain an ISOAPHeaders (see page 923) interface instead of obtaining one from the implementation class.

See Also

InvRegistry (see page 930)

IInvokable

TInvokableClassRegistry (see page 840)

THTTPSoapPascalInvoker

ISOAPHeaders (see page 923)

Adding New Web Services

23.1.6.1 TInvokableClass Members

The following tables list the members exposed by TInvokableClass.

Protected Fields

	Name	Description
	FRefCount (see page 836)	This is FRefCount, a member of class TInvokableClass.
	FSOAPHeaders (see page 836)	This is FSOAPHeaders, a member of class TInvokableClass.

Protected Methods

	Name	Description
	_AddRef (see page 836)	Implements the IInterface _AddRef method.
	_Release (see page 837)	Implements the IInterface _Release method.
	QueryInterface (see page 839)	Implements the IInterface QueryInterface method.

Public Methods

	Name	Description
V	AfterConstruction (see page 837)	Responds after the last constructor executes.
V	BeforeDestruction (see page 838)	Checks that the reference count is zero before allowing the interfaced object to be destroyed.
V	Create (see page 835)	Creates and initializes a TInvokableClass (see page 833) object.
V	NewInstance (see page 838)	Allocates memory for each instance of an invokable object.

Public Properties

	Name	Description
R	RefCount (see page 839)	Indicates the number of interface pointers currently dependent upon the object.

23.1.6.2 TInvokableClass.Create Constructor

Creates and initializes a TInvokableClass (see page 833) object.

Pascal

```
constructor Create; virtual;
```

C++

```
virtual __fastcall TInvokableClass();
```

Description

The TInvokableClass (see page 833) constructor does nothing special. It exists primarily because

The TInvokableClass (see page 833) constructor exists primarily because

The invocation registry knows how to call it.

It is virtual, so that the invocation registry can also instantiate descendant classes.

In addition, the constructor initializes the invokable class so that it can support the ISOAPHeaders (see page 923) interface.

See Also

InvRegistry (see page 930)

ISOAPHeaders (see page 923)

23.1.6.3 TInvokableClass Fields

The fields of the TInvokableClass class are listed here.

Protected Fields

	Name	Description
	FRefCount (see page 836)	This is FRefCount, a member of class TInvokableClass.
	FSOAPHeaders (see page 836)	This is FSOAPHeaders, a member of class TInvokableClass.

23.1.6.3.1 TInvokableClass.FRefCount Field

Pascal

```
FRefCount: Integer;
```

C++

```
int FRefCount;
```

Description

This is FRefCount, a member of class TInvokableClass.

23.1.6.3.2 TInvokableClass.FSOAPHeaders Field

Pascal

```
FSOAPHeaders: TSOAPHeaders;
```

C++

```
TSOAPHeaders FSOAPHeaders;
```

Description

This is FSOAPHeaders, a member of class TInvokableClass.

23.1.6.4 TInvokableClass Methods

The methods of the TInvokableClass class are listed here.

Protected Methods

	Name	Description
	_AddRef (see page 836)	Implements the IInterface _AddRef method.
	_Release (see page 837)	Implements the IInterface _Release method.
	QueryInterface (see page 839)	Implements the IInterface QueryInterface method.

Public Methods

	Name	Description
	AfterConstruction (see page 837)	Responds after the last constructor executes.
	BeforeDestruction (see page 838)	Checks that the reference count is zero before allowing the interfaced object to be destroyed.
	NewInstance (see page 838)	Allocates memory for each instance of an invokable object.

23.1.6.4.1 TInvokableClass._AddRef Method

Implements the IInterface _AddRef method.

Pascal

```
function _AddRef: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _AddRef();
```

Description

Do not call the protected __AddRef method directly. __AddRef is called through the interface to indicate that another object is using the invokable object's interface pointer. When the other object is through with the interface, it calls _Release (see page 837) through the same interface. This allows the invokable object to free itself when it is no longer used (when the reference count goes to zero).

__AddRef increments the RefCount (see page 839) property.

See Also

RefCount (see page 839)

_Release (see page 837)

23.1.6.4.2 TInvokableClass._Release Method

Implements the IInterface _Release method.

Pascal

```
function _Release: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _Release();
```

Description

Do not call the protected __Release method directly. __Release is called through the IInterface interface to indicate that another object is through with the interfaced object's interface pointer. This allows the interfaced object to free itself when it is no longer used (when the reference count goes to zero).

__Release decrements the RefCount (see page 839) property.

See Also

RefCount (see page 839)

_AddRef (see page 836)

BeforeDestruction (see page 838)

23.1.6.4.3 TInvokableClass.AfterConstruction Method

Responds after the last constructor executes.

Pascal

```
procedure AfterConstruction; override;
```

C++

```
virtual __fastcall AfterConstruction();
```

Description

AfterConstruction is called automatically after the object's last constructor has executed. Do not call it explicitly in your applications.

When an instance of TInvokableClass (see page 833) is allocated, RefCount (see page 839) is incremented to prevent its

destruction while processing the constructor. AfterConstruction decrements that RefCount (see page 839) to re-establish the correct reference count after all constructors have executed.

See Also

BeforeDestruction (see page 838)

NewInstance (see page 838)

RefCount (see page 839)

23.1.6.4.4 TInvokableClass.BeforeDestruction Method

Checks that the reference count is zero before allowing the interfaced object to be destroyed.

Pascal

```
procedure BeforeDestruction; override;
```

C++

```
virtual __fastcall BeforeDestruction();
```

Description

BeforeDestruction is called automatically immediately before the interfaced object's destructor. It checks that the reference count is zero, and, if it is not, raises an exception.

See Also

RefCount (see page 839)

System::TObject::Free

23.1.6.4.5 TInvokableClass.NewInstance Method

Allocates memory for each instance of an invokable object.

Pascal

```
class function NewInstance: TObject; override;
```

C++

```
virtual __fastcall TObject * NewInstance();
```

Description

Constructors call NewInstance automatically. NewInstance calls InstanceSize to determine how much memory to allocate from the heap to contain a particular instance. Do not call NewInstance directly.

TInvokableClass (see page 833) overrides NewInstance to increment the reference count when a new instance is created. This prevents any constructor from accidentally deleting the instance due to a RefCount (see page 839) of zero. The AfterConstruction (see page 837) method decrements the value of RefCount (see page 839) after all constructors have executed.

See Also

RefCount (see page 839)

System::TObject::NewInstance

System::TObject::InstanceSize

AfterConstruction (see page 837)

23.1.6.4.6 TInvokableClass.QueryInterface Method

Implements the IInterface QueryInterface method.

Pascal

```
function QueryInterface(const IID: TGUID; out Obj): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT QueryInterface(const TGUID IID,  Obj);
```

Description

Do not call the protected QueryInterface method directly. QueryInterface is called through the IInterface interface to obtain an interface pointer for the interface identified by the IID parameter. If the object supports the requested interface, it is returned in the Obj parameter and QueryInterface returns S_OK. If the object does not support the interface, QueryInterface returns E_NOINTERFACE.

If QueryInterface successfully returns an interface pointer, it automatically increments the reference count.

See Also

_AddRef (see page 836)

23.1.6.5 TInvokableClass Properties

The properties of the TInvokableClass class are listed here.

Public Properties

	Name	Description
	RefCount (see page 839)	Indicates the number of interface pointers currently dependent upon the object.

23.1.6.5.1 TInvokableClass.RefCount Property

Indicates the number of interface pointers currently dependent upon the object.

Pascal

```
property RefCount: Integer;
```

C++

```
__property int RefCount;
```

Description

RefCount defines the lifetime of the object. When RefCount is zero, the object is destroyed.

RefCount is incremented by calls to the _AddRef (see page 836) method and decremented by the _Release (see page 837) method.

Note:

See Also

_AddRef (see page 836)

_Release (see page 837)

23.1.7 TInvokableClassRegistry Class

TInvokableClassRegistry is the type of the invocation registry.

Class Hierarchy

Pascal

```
TInvokableClassRegistry = class(TInterfacedObject);
```

C++

```
class TInvokableClassRegistry : public TInterfacedObject;
```

File

InvokeRegistry

Description

TInvokableClassRegistry manages information about invokable interfaces and the classes that implement them. Invokable interfaces are used in applications that work with Web Services.

Server applications that implement Web Services need to register both the invokable interfaces and the implementation classes. The invoker component (such as THTTPSoapPascalInvoker or THTTPSoapCppInvoker) can then use the invocation registry to locate invokable interfaces and execute calls made to them. In addition, server applications should register the headers that are used with calls to the methods of the invokable interface and the exceptions that such calls can raise.

Client applications that call on Web Services register the invokable interfaces that they call. The remote interfaced object component (THTTPRio) can then use the invocation registry to generate a dynamic vtable for calling the invokable interfaces.

Do not create instances of TInvokableClassRegistry. Instead, use the global InvRegistry (see page 930) function to access the invocation registry. Most applications need only call the RegisterInterface (see page 864) and (if a server) the RegisterInvokableClass (see page 865) methods. The other TInvokableClassRegistry methods are used by invoker components, remote interfaced object components, and the WSDL document importer. The allow these components to look up information about invokable interfaces and their implementation classes, access invokable interfaces, and so on.

See Also

InvRegistry (see page 930)

IInvokable

THTTPSoapCppInvoker

THTTPSoapPascalInvoker

THTTPRIO

23.1.7.1 TInvokableClassRegistry Members

The following tables list the members exposed by TInvokableClassRegistry.

Public Methods

	Name	Description
	Create (see page 842)	Creates and initializes a TInvokableClassRegistry (see page 840) object.

	Destroy (see page 842)	Frees the memory associated with the TInvokableClassRegistry (see page 840) object.
	GetActionURIOfIID (see page 844)	Returns the SOAP Action header associated with the interface with a specified GUID.
	GetActionURIOfInfo (see page 844)	Returns the SOAP Action header associated with the type info of a registered interface.
	GetClassFromIntfInfo (see page 845)	Retrieves the implementation class that corresponds to the type info of an invokable interface.
	GetExceptionInfoForInterface (see page 845)	Retrieves an array of exceptions that are registered as associated with an invokable interface.
	GetHeaderClass (see page 846)	Returns the class reference for a header type, given its name and namespace.
	GetHeaderInfoForInterface (see page 846)	Retrieves an array of headers that are registered as associated with an invokable interface.
	GetHeaderName (see page 847)	Returns the name of a header that is identified by a specified class reference.
	GetHeaderNamespace (see page 847)	This is the overview for the GetHeaderNamespace method overload.
	GetInfoForURI (see page 848)	Retrieves information about a registered invokable interface that is identified by a SOAP action header or the path of an HTTP message.
	GetInterfaceCount (see page 849)	Returns the number of registered invokable interfaces.
	GetInterfaceExternalName (see page 849)	This is the overview for the GetInterfaceExternalName method overload.
	GetInterfaceInfoFromName (see page 850)	Returns information about a registered invokable interface given its name.
	GetInterfaceTypeInfo (see page 850)	Returns the runtime type information (RTTI) of an invokable interface.
	GetIntfInvokeOptions (see page 851)	This is the overview for the GetIntfInvokeOptions method overload.
	GetInvokableObjectFromClass (see page 852)	Returns an instance of the implementation class for a registered invokable interface.
	GetMethExternalName (see page 852)	Returns the name of the operation that corresponds to a method on an invokable interface.
	GetMethInternalName (see page 852)	Returns the name of the method that corresponds to an operation in a WSDL document.
	GetNamespaceByGUID (see page 853)	Returns a namespace that is uniquely associated with a registered interface.
	GetParamExternalName (see page 853)	Returns the name of the part that corresponds to a parameter on a method of an invokable interface.
	GetParamInternalName (see page 854)	Returns the name of the parameter that corresponds to a part on a message of an operation in a WSDL document.
	GetRegInterfaceEntry (see page 854)	Returns the internal data that TInvokableClassRegistry (see page 840) maintains for one of its registered interfaces.
	GetRequestHeaderInfoForInterface (see page 855)	Retrieves an array of headers that are registered as associated with requests to an invokable interface.
	GetResponseHeaderInfoForInterface (see page 855)	Retrieves an array of headers that are registered as associated with responses to invocations of an invokable interface.
	GetReturnParamNames (see page 856)	Returns the list of parameter names that map to return values of interface methods.
	GetUDDIInfo (see page 856)	This is the overview for the GetUDDIInfo method overload.
	GetWSDLEncoding (see page 857)	This is the overview for the GetWSDLEncoding method overload.
	HasRegInterfaceImpl (see page 858)	Indicates whether there is a registered implementation class for an invokable interface.
	RegisterAllSOAPActions (see page 859)	Supplies a set of SOAP Action headers to use for an already-registered interface.
	RegisterDefaultSOAPAction (see page 859)	Supplies a SOAP Action header to use for an already-registered interface.
	RegisterException (see page 860)	Registers the remotable exception classes that calls to an invokable interface can raise.
	RegisterExceptionMethod (see page 860)	Registers methods of an invokable interface that can raise a registered exception class.
	RegisterExternalMethName (see page 861)	Registers the correspondence between a method name on an invokable interface and the corresponding operation name in a WSDL document.
	RegisterExternalParamName (see page 861)	Registers the correspondence between a parameter name on an invokable interface and the corresponding part name in a WSDL document.
	RegisterHeaderClass (see page 862)	This is the overview for the RegisterHeaderClass method overload.
	RegisterHeaderMethod (see page 863)	Registers methods of an invokable interface that make use of a registered header.
	RegisterInterface (see page 864)	Registers an invokable interface.
	RegisterInvokableClass (see page 865)	This is the overview for the RegisterInvokableClass method overload.
	RegisterInvokeOptions (see page 866)	This is the overview for the RegisterInvokeOptions method overload.

	RegisterReturnParamNames (see page 867)	Registers a list of parameter names that map to return values of interface methods.
	RegisterUDDIInfo (see page 867)	Registers the location of the UDDI description that can be used to import the location of a Web Service.
	UnRegisterInterface (see page 867)	Unregisters an invokable interface.
	UnRegisterInvokableClass (see page 868)	Unregisters the implementation class for an invokable interface.

23.1.7.2 TInvokableClassRegistry.Create Constructor

Creates and initializes a TInvokableClassRegistry (see page 840) object.

Pascal

```
constructor Create;
```

C++

```
__fastcall TInvokableClassRegistry();
```

Description

Do not create your own instances of TInvokableClassRegistry (see page 840). Instead, use the global InvRegistry (see page 930) function, which provides access to a single global instance of TInvokableClassRegistry (see page 840).

See Also

InvRegistry (see page 930)

23.1.7.3 TInvokableClassRegistry.Destroy Destructor

Frees the memory associated with the TInvokableClassRegistry (see page 840) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TInvokableClassRegistry();
```

Description

Do not destroy the global TInvokableClassRegistry (see page 840) instance that is available through the InvRegistry (see page 930) function. This object is freed automatically when the application unloads the InvokeRegistry (see page 807) unit.

See Also

InvRegistry (see page 930)

23.1.7.4 TInvokableClassRegistry Methods

The methods of the TInvokableClassRegistry class are listed here.

Public Methods

	Name	Description
	GetActionURIOfIID (see page 844)	Returns the SOAP Action header associated with the interface with a specified GUID.
	GetActionURIOfInfo (see page 844)	Returns the SOAP Action header associated with the type info of a registered interface.
	GetClassFromIntfInfo (see page 845)	Retrieves the implementation class that corresponds to the type info of an invokable interface.

	GetExceptionInfoForInterface (see page 845)	Retrieves an array of exceptions that are registered as associated with an invokable interface.
	GetHeaderClass (see page 846)	Returns the class reference for a header type, given its name and namespace.
	GetHeaderInfoForInterface (see page 846)	Retrieves an array of headers that are registered as associated with an invokable interface.
	GetHeaderName (see page 847)	Returns the name of a header that is identified by a specified class reference.
	GetHeaderNamespace (see page 847)	This is the overview for the GetHeaderNamespace method overload.
	GetInfoForURI (see page 848)	Retrieves information about a registered invokable interface that is identified by a SOAP action header or the path of an HTTP message.
	GetInterfaceCount (see page 849)	Returns the number of registered invokable interfaces.
	GetInterfaceExternalName (see page 849)	This is the overview for the GetInterfaceExternalName method overload.
	GetInterfaceInfoFromName (see page 850)	Returns information about a registered invokable interface given its name.
	GetInterfaceTypeInfo (see page 850)	Returns the runtime type information (RTTI) of an invokable interface.
	GetIntfInvokeOptions (see page 851)	This is the overview for the GetIntfInvokeOptions method overload.
	GetInvokableObjectFromClass (see page 852)	Returns an instance of the implementation class for a registered invokable interface.
	GetMethExternalName (see page 852)	Returns the name of the operation that corresponds to a method on an invokable interface.
	GetMethInternalName (see page 852)	Returns the name of the method that corresponds to an operation in a WSDL document.
	GetNamespaceByGUID (see page 853)	Returns a namespace that is uniquely associated with a registered interface.
	GetParamExternalName (see page 853)	Returns the name of the part that corresponds to a parameter on a method of an invokable interface.
	GetParamInternalName (see page 854)	Returns the name of the parameter that corresponds to a part on a message of an operation in a WSDL document.
	GetRegInterfaceEntry (see page 854)	Returns the internal data that TInvokableClassRegistry (see page 840) maintains for one of its registered interfaces.
	GetRequestHeaderInfoForInterface (see page 855)	Retrieves an array of headers that are registered as associated with requests to an invokable interface.
	GetResponseHeaderInfoForInterface (see page 855)	Retrieves an array of headers that are registered as associated with responses to invocations of an invokable interface.
	GetReturnParamNames (see page 856)	Returns the list of parameter names that map to return values of interface methods.
	GetUDDIInfo (see page 856)	This is the overview for the GetUDDIInfo method overload.
	GetWSDLEncoding (see page 857)	This is the overview for the GetWSDLEncoding method overload.
	HasRegInterfaceImpl (see page 858)	Indicates whether there is a registered implementation class for an invokable interface.
	RegisterAllSOAPActions (see page 859)	Supplies a set of SOAP Action headers to use for an already-registered interface.
	RegisterDefaultSOAPAction (see page 859)	Supplies a SOAP Action header to use for an already-registered interface.
	RegisterException (see page 860)	Registers the remotable exception classes that calls to an invokable interface can raise.
	RegisterExceptionMethod (see page 860)	Registers methods of an invokable interface that can raise a registered exception class.
	RegisterExternalMethName (see page 861)	Registers the correspondence between a method name on an invokable interface and the corresponding operation name in a WSDL document.
	RegisterExternalParamName (see page 861)	Registers the correspondence between a parameter name on an invokable interface and the corresponding part name in a WSDL document.
	RegisterHeaderClass (see page 862)	This is the overview for the RegisterHeaderClass method overload.
	RegisterHeaderMethod (see page 863)	Registers methods of an invokable interface that make use of a registered header.
	RegisterInterface (see page 864)	Registers an invokable interface.
	RegisterInvokableClass (see page 865)	This is the overview for the RegisterInvokableClass method overload.
	RegisterInvokeOptions (see page 866)	This is the overview for the RegisterInvokeOptions method overload.
	RegisterReturnParamNames (see page 867)	Registers a list of parameter names that map to return values of interface methods.
	RegisterUDDIInfo (see page 867)	Registers the location of the UDDI description that can be used to import the location of a Web Service.
	UnRegisterInterface (see page 867)	Unregisters an invokable interface.
	UnRegisterInvokableClass (see page 868)	Unregisters the implementation class for an invokable interface.

23.1.7.4.1 TInvokableClassRegistry.GetActionURIOfIID Method

Returns the SOAP Action header associated with the interface with a specified GUID.

Pascal

```
function GetActionURIOfIID(const AGUID: TGUID): string;
```

C++

```
__fastcall UnicodeString GetActionURIOfIID(const TGUID AGUID);
```

Description

Typically, applications do not need to call GetActionURIOfIID. This method is called by remote interfaced objects (THTTPRIO) when generating the SOAP Action header of messages that call a method on a registered interface.

AGUID is the globally unique identifier (GUID) of the interface to call. The interface must be previously registered by a call to the RegisterInterface (see page 864) method.

GetActionURIOfIID returns the string that should appear as the SOAP Action header on request messages to that interface.

See Also

GetActionURIOfInfo (see page 844)

RegisterDefaultSOAPAction (see page 859)

RegisterInterface (see page 864)

THTTPRIO

23.1.7.4.2 TInvokableClassRegistry.GetActionURIOfInfo Method

Returns the SOAP Action header associated with the type info of a registered interface.

Pascal

```
function GetActionURIOfInfo(const IntfInfo: PTypeInfo; const MethodName: InvString;
MethodIndex: Integer): string;
```

C++

```
__fastcall UnicodeString GetActionURIOfInfo(const PTypeInfo IntfInfo, const InvString
MethodName, int MethodIndex);
```

Description

GetActionURIOfInfo looks up or generates the SOAP Action header that should be used with messages that call a specified method on an invokable interface.

IntfInfo (see page 789) is the runtime type information (RTTI) that is compiled for the invokable interface. The interface must have been previously registered by a call to the RegisterInterface (see page 864) method.

MethodName is the name of the method to call on the specified interface.

MethodIndex indicates which method to use when the interface includes an overload of the method that MethodName specifies. 0 indicates the overload that is declared first, 1 the overload that is declared second, and so on.

GetActionURIOfInfo returns the string that should appear as the SOAP Action header on request messages to the specified method of the specified interface.

See Also

GetActionURIOfIID (see page 844)

RegisterDefaultSOAPAction (see page 859)

RegisterInterface (see page 864)

THTTPRIO

23.1.7.4.3 TInvokableClassRegistry.GetClassFromIntfInfo Method

Retrieves the implementation class that corresponds to the type info of an invokable interface.

Pascal

```
procedure GetClassFromIntfInfo(Info: PTypeInfo; var AClass: TClass);
```

C++

```
__fastcall GetClassFromIntfInfo(PTypeInfo Info, TClass AClass);
```

Description

Typically, applications do not need to call GetClassFromIntfInfo. This method allows the invoker component (THTTPSoapPascalInvoker or THTTPSoapCppInvoker) to look up the implementation class for an interface it is invoking.

Info is the runtime type information (RTTI) from a registered invokable interface. The interface must have been previously registered by a call to the RegisterInterface (see page 864) method.

AClass returns a class reference for the class that implements the invokable interface identified by Info. It can be passed to the GetInvokableObjectFromClass (see page 852) method to obtain an instance of the implementation class.

See Also

GetInvokableObjectFromClass (see page 852)

RegisterInvokableClass (see page 865)

RegisterInterface (see page 864)

THTTPSoapPascalInvoker

THTTPSoapCppInvoker

23.1.7.4.4 TInvokableClassRegistry.GetExceptionInfoForInterface Method

Retrieves an array of exceptions that are registered as associated with an invokable interface.

Pascal

```
function GetExceptionInfoForInterface(Info: PTypeInfo): TExceptionItemArray;
```

C++

```
__fastcall TExceptionItemArray GetExceptionInfoForInterface(PTypeInfo Info);
```

Description

Typically, applications do not need to call GetExceptionInfoForInterface. This method is used internally by the WSDL publisher to generate the WSDL that describes the exceptions that an invokable interface can raise.

Info is the runtime type information (RTTI) from a registered invokable interface. The interface must have been previously registered by a call to the RegisterInterface (see page 864) method.

GetExceptionInfoForInterface returns an array of IntfExceptionItem (see page 935) values, each of which describes a remotable exception that was registered for the invokable interface by a call to the RegisterException (see page 860) method.

See Also

RegisterException (see page 860)

TWSDLHTMLPublish

RegisterInterface (see page 864)

23.1.7.4.5 TInvokableClassRegistry.GetHeaderClass Method

Returns the class reference for a header type, given its name and namespace.

Pascal

```
function GetHeaderClass(Name: InvString; Namespace: InvString): TClass;
```

C++

```
__fastcall TClass GetHeaderClass(InvString Name, InvString Namespace);
```

Description

Typically, applications do not need to call GetHeaderClass. This method is used internally to look up the class of a header that is identified by name.

Name is the name of the header node.

Namespace is the URI of the namespace in which Name is defined.

GetHeaderClass returns the class reference for the TSOAPHeader (see page 908) descendant that represents the identified header.

See Also

RegisterHeaderClass (see page 862)

TSOAPHeader (see page 908)

GetHeaderName (see page 847)

GetHeaderNamespace (see page 848)

23.1.7.4.6 TInvokableClassRegistry.GetHeaderInfoForInterface Method

Retrieves an array of headers that are registered as associated with an invokable interface.

Pascal

```
function GetHeaderInfoForInterface(Info: PTypeInfo; MethodType: eHeaderMethodType = hmtAll):
THeaderItemArray;
```

C++

```
__fastcall THeaderItemArray GetHeaderInfoForInterface(PTypeInfo Info, eHeaderMethodType
MethodType = hmtAll);
```

Description

Typically, applications do not need to call GetHeaderInfoForInterface. This method is used internally by the WSDL publisher to generate the WSDL that describes the headers that are used in messages about an invokable interface.

Info is the runtime type information (RTTI) from a registered invokable interface. The interface must have been previously registered by a call to the RegisterInterface (see page 864) method.

MethodType indicates whether to retrieve information about the headers to use on request messages, the headers to use on

response messages, or both.

GetHeaderInfoForInterface returns an array of IntfHeaderItem (see page 936) values, each of which describes a header that was registered for the invokable interface by a call to the RegisterHeaderClass (see page 862) method.

See Also

RegisterHeaderClass (see page 862)

TWSDLHTMLPublish

RegisterInterface (see page 864)

23.1.7.4.7 TInvokableClassRegistry.GetHeaderName Method

Returns the name of a header that is identified by a specified class reference.

Pascal

```
function GetHeaderName(Info: PTypeInfo; AClass: TClass): InvString;
```

C++

```
__fastcall InvString GetHeaderName(PTypeInfo Info, TClass AClass);
```

Description

GetHeaderName returns the name of a header given the class that represents it. This is the name supplied to a call to RegisterHeaderClass (see page 862) when the header was registered. If the header was registered without supplying a name, this is the same name as the type name supplied for the header when it is registered with the remotable type registry.

Info is the runtime type information for the interface associated with the header.

AClass is a class reference to the TSOAPHeader (see page 908) descendant that represents the header.

GetHeaderName returns the name of the header. This name is defined within the namespace that the GetHeaderNamespace (see page 848) method returns.

See Also

RegisterHeaderClass (see page 862)

TSOAPHeader (see page 908)

GetHeaderNamespace (see page 848)

GetHeaderClass (see page 846)

23.1.7.4.8 GetHeaderNamespace Method

This is the overview for the GetHeaderNamespace method overload.

Overload List

	Name	Description
	TInvokableClassRegistry.GetHeaderNamespace (PTypeInfo, TClass) (see page 848)	Returns the namespace URI for a header that is identified by a specified class reference.
	TInvokableClassRegistry.GetHeaderNamespace (TClass) (see page 848)	Returns the namespace URI for a header that is identified by a specified class reference.

23.1.7.4.8.1 TInvokableClassRegistry.GetHeaderNamespace Method (PTypeInfo, TClass)

Returns the namespace URI for a header that is identified by a specified class reference.

Pascal

```
function GetHeaderNamespace(Info: PTypeInfo; AClass: TClass): InvString; overload;
function GetHeaderNamespace(AClass: TClass): InvString; overload;
```

C++

```
__fastcall InvString GetHeaderNamespace(PTypeInfo Info, TClass AClass);
__fastcall InvString GetHeaderNamespace(TClass AClass);
```

Description

GetHeaderNamespace returns the namespace in which a header's name is defined. This is the namespace URI supplied to a call to RegisterHeaderClass (see page 862) when the header was registered. If the header was registered without supplying a namespace URI, this is the same namespace as the one associated with the interface that uses the header.

Info is the runtime type information for the interface associated with the header.

AClass is a class reference to the TSOAPHeader (see page 908) descendant that represents the header.

GetHeaderNamespace returns the namespace URI to use for resolving the name of the header.

See Also

RegisterHeaderClass (see page 862)

TSOAPHeader (see page 908)

GetHeaderName (see page 847)

GetHeaderClass (see page 846)

23.1.7.4.9 TInvokableClassRegistry.GetInfoForURI Method

Retrieves information about a registered invokable interface that is identified by a SOAP action header or the path of an HTTP message.

Pascal

```
function GetInfoForURI(const PathURI: string; const ActionURI: string; var ACLass: TClass; var
IntfInfo: PTypeInfo; var AMeth: string): Boolean;
```

C++

```
__fastcall Boolean GetInfoForURI(const UnicodeString PathURI, const UnicodeString ActionURI,
TClass ACLass, PTypeInfo IntfInfo, UnicodeString AMeth);
```

Description

Typically, applications do not need to call GetInfoForURI. This method is called by the invoker component (THTTPSoapPascalInvoker or THTTPSoapCppInvoker) to look up the target of a SOAP message that it is invoking.

PathURI is the path portion of the target URI on the SOAP message. When using a path to identify an invokable interface, the path portion must end in the name of the unit followed by the name of the interface. It is only used if ActionURI is an empty string.

ActionURI is the SOAP action header of the SOAP message. It identifies an interface based on the namespace in the SOAP action header. This namespace must match the namespace supplied to the RegisterInterface (see page 864) method (or generated by that method if the caller did not supply a namespace). If ActionURI is an empty string, GetInfoForURI uses PathURI instead to identify the target interface.

AClass returns a class reference for the class that implements the invokable interface identified by PathURI or ActionURI.

IntfInfo (see page 789) returns the runtime type information (RTTI) of the invokable interface identified by PathURI or ActionURI.

AMeth is not currently used.

See Also

DispatchSOAP

RegisterInterface (see page 864)

23.1.7.4.10 TInvokableClassRegistry.GetInterfaceCount Method

Returns the number of registered invokable interfaces.

Pascal

```
function GetInterfaceCount: Integer;
```

C++

```
__fastcall int GetInterfaceCount();
```

Description

GetInterfaceCount returns the number of invokable interfaces that have been registered. Each invokable interface is registered by a call to the RegisterInterface (see page 864) method.

See Also

RegisterInterface (see page 864)

23.1.7.4.11 GetInterfaceExternalName Method

This is the overview for the GetInterfaceExternalName method overload.

Overload List

	Name	Description
	TInvokableClassRegistry.GetInterfaceExternalName (PTypeInfo) (see page 849)	Returns the name of the port type that corresponds to an invokable interface.
	TInvokableClassRegistry.GetInterfaceExternalName (PTypeInfo, string) (see page 849)	Returns the name of the port type that corresponds to an invokable interface.
	TInvokableClassRegistry.GetInterfaceExternalName (PTypeInfo, string, string) (see page 849)	Returns the name of the port type that corresponds to an invokable interface.

23.1.7.4.11.1 TInvokableClassRegistry.GetInterfaceExternalName Method (PTypeInfo)

Returns the name of the port type that corresponds to an invokable interface.

Pascal

```
function GetInterfaceExternalName(Info: PTypeInfo; const Namespace: string; const
InternalIntfName: string): InvString; overload;
function GetInterfaceExternalName(Info: PTypeInfo; const Namespace: string): InvString;
overload;
function GetInterfaceExternalName(Info: PTypeInfo): InvString; overload;
```

C++

```
__fastcall InvString GetInterfaceExternalName(PTypeInfo Info, const UnicodeString Namespace,
const UnicodeString InternalIntfName);
```

```
__fastcall InvString GetInterfaceExternalName(PTypeInfo Info, const UnicodeString Namespace);
__fastcall InvString GetInterfaceExternalName(PTypeInfo Info);
```

Description

GetInterfaceExternalName returns the name of the port type that corresponds to a specified invokable interface. This is the name given to the corresponding port type in a WSDL document. The association between the external name and the interface name is established by the RegisterInterface (see page 864) method.

Info is the type information pointer for the invokable interface.

Namespace is the namespace assigned to the invokable interface when it was registered by a call to the RegisterInterface (see page 864) method. If the interface was registered using the automatically-generated namespace, this parameter can be omitted or you can supply an empty string.

InternalIntfName is the name of the invokable interface.

See Also

RegisterInterface (see page 864)

23.1.7.4.12 TInvokableClassRegistry.GetInterfaceInfoFromName Method

Returns information about a registered invokable interface given its name.

Pascal

```
procedure GetInterfaceInfoFromName(const UnitName: string; const IntfName: string; var Info:
PTypeInfo; var IID: TGUID);
```

C++

```
__fastcall GetInterfaceInfoFromName(const UnicodeString UnitName, const UnicodeString
IntfName, PTypeInfo Info, TGUID IID);
```

Description

GetInterfaceInfoFromName looks up a registered interface given its name and the name of the unit in which it is declared.

UnitName is the name of the unit that contains the interface definition.

IntfName is the name of the interface.

Info returns the type information that is compiled with the specified interface.

IID returns the globally unique identifier for the specified interface.

See Also

GetInterfaceTypeInfo (see page 850)

GetInfoForURI (see page 848)

23.1.7.4.13 TInvokableClassRegistry.GetInterfaceTypeInfo Method

Returns the runtime type information (RTTI) of an invokable interface.

Pascal

```
function GetInterfaceTypeInfo(const AGUID: TGUID): Pointer;
```

C++

```
__fastcall void * GetInterfaceTypeInfo(const TGUID AGUID);
```

Description

GetInterfaceTypeInfo returns a pointer to the RTTI that is compiled with an invokable interface.

AGUID is the globally unique identifier that specifies the desired interface.

See Also

GetInterfaceInfoFromName (see page 850)

GetInfoForURI (see page 848)

23.1.7.4.14 GetIntfInvokeOptions Method

This is the overview for the GetIntfInvokeOptions method overload.

Overload List

	Name	Description
	TInvokableClassRegistry.GetIntfInvokeOptions (PTypeInfo) (see page 851)	Retrieves any flags that customize the way calls to a registered interface are made.
	TInvokableClassRegistry.GetIntfInvokeOptions (TGUID) (see page 851)	Retrieves any flags that customize the way calls to a registered interface are made.

23.1.7.4.14.1 TInvokableClassRegistry.GetIntfInvokeOptions Method (PTypeInfo)

Retrieves any flags that customize the way calls to a registered interface are made.

Pascal

```
function GetIntfInvokeOptions(const IntfInfo: PTypeInfo): TIntfInvokeOptions; overload;
function GetIntfInvokeOptions(const AGUID: TGUID): TIntfInvokeOptions; overload;
```

C++

```
__fastcall TIntfInvokeOptions GetIntfInvokeOptions(const PTypeInfo IntfInfo);
__fastcall TIntfInvokeOptions GetIntfInvokeOptions(const TGUID AGUID);
```

Description

GetIntfInvokeOptions retrieves any invocation options that have been registered for an interface using the RegisterInvokeOptions (see page 866) method. Invocation options customize the way a registered interface is called, especially the way such calls are marshaled or unmarshaled. Remote interfaced objects (THTTPRIO) call this method to determine what options they need to set on the component that is the value of their Converter property.

AGUID identifies the registered interface using its GUID.

IntfInfo (see page 789) identifies the registered interface using its runtime type information (RTTI). You can obtain the type information from an invokable interface by calling the TypeInfo function (delphi) or the __delphirtti operator (C++).

GetIntfInvokeOptions returns a set of flags that indicate how to make calls to the specified interface.

See Also

RegisterInvokeOptions (see page 866)

Converter

Options (see page 2781)

RegisterInterface (see page 864)

23.1.7.4.15 TInvokableClassRegistry.GetInvokableObjectFromClass Method

Returns an instance of the implementation class for a registered invokable interface.

Pascal

```
function GetInvokableObjectFromClass(AClass: TClass): TObject;
```

C++

```
__fastcall TObject * GetInvokableObjectFromClass(TClass AClass);
```

Description

Typically, applications do not need to call GetInvokableObjectFromClass. This method is called by the invoker component (THTTPSoapPascalInvoker or THTTPSoapCppInvoker) to obtain an instance of the class that implements a registered invokable interface. When the invoker is finished with the object, it calls GetWSDLEncoding (see page 857) to free the instance.

AClass identifies the class whose instance is desired. It must be registered by a call to RegisterInvokableClass (see page 865). The invoker obtains this class reference by calling the GetInfoForURI (see page 848) method.

See Also

GetInfoForURI (see page 848)

RegisterInvokableClass (see page 865)

GetWSDLEncoding (see page 857)

23.1.7.4.16 TInvokableClassRegistry.GetMethExternalName Method

Returns the name of the operation that corresponds to a method on an invokable interface.

Pascal

```
function GetMethExternalName(Info: PTypeInfo; const MethodIntName: string): InvString;
```

C++

```
__fastcall InvString GetMethExternalName(PTypeInfo Info, const UnicodeString MethodIntName);
```

Description

GetMethExternalName returns the name of the operation that corresponds to a specified method. This is the name used to define the operation for that method in a WSDL document. The association between the external name and the method name is established by the RegisterExternalMethName (see page 861) method.

Info is the type information pointer for the invokable interface that includes the method.

MethodIntName is the name of the method on that invokable interface.

See Also

RegisterExternalMethName (see page 861)

GetMethInternalName (see page 852)

23.1.7.4.17 TInvokableClassRegistry.GetMethInternalName Method

Returns the name of the method that corresponds to an operation in a WSDL document.

Pascal

```
function GetMethInternalName(Info: PTypeInfo; const MethodExtName: InvString): string;
```

C++

```
__fastcall UnicodeString GetMethInternalName(PTypeInfo Info, const InvString MethodExtName);
```

Description

GetMethInternalName returns the name of a method on a registered invokable interface that corresponds to a specified operation. The operation name comes from a WSDL document that describes the invokable interface and its methods. The association between the operation name and the method name is established by the RegisterExternalMethName (see page 861) method.

Info is the type information pointer for the invokable interface that includes the method.

MethodExtName is the name of the operation for which you want the corresponding method name.

See Also

RegisterExternalMethName (see page 861)

GetMethExternalName (see page 852)

23.1.7.4.18 TInvokableClassRegistry.GetNamespaceByGUID Method

Returns a namespace that is uniquely associated with a registered interface.

Pascal

```
function GetNamespaceByGUID(const AGUID: TGUID): string;
```

C++

```
__fastcall UnicodeString GetNamespaceByGUID(const TGUID AGUID);
```

Description

GetNamespaceByGUID returns the namespace URI that identifies the specified interface. The namespace is originally associated with the interface when it is registered by a call to the RegisterInterface (see page 864) method. This namespace can be supplied when the interface is registered, or automatically generated by the RegisterInterface (see page 864) method.

AGUID is the globally unique identifier that identifies the interface whose namespace is desired..

See Also

GetInfoForURI (see page 848)

RegisterInterface (see page 864)

23.1.7.4.19 TInvokableClassRegistry.GetParamExternalName Method

Returns the name of the part that corresponds to a parameter on a method of an invokable interface.

Pascal

```
function GetParamExternalName(Info: PTypeInfo; const MethodName: string; const
InternalParamName: string): InvString;
```

C++

```
__fastcall InvString GetParamExternalName(PTypeInfo Info, const UnicodeString MethodName,
const UnicodeString InternalParamName);
```

Description

GetParamExternalName returns the name of the part that corresponds to a specified method parameter. This is the name used to define the part for that parameter in a WSDL document. The association between the external name and the parameter name

is established by the RegisterExternalParamName (see page 861) method.

Info is the type information pointer for the invokable interface that includes the method.

MethodName is the name of the method on that invokable interface.

InternalParamName is the name of the parameter on the method of the invokable interface.

See Also

RegisterExternalParamName (see page 861)

GetParamInternalName (see page 854)

23.1.7.4.20 TInvokableClassRegistry.GetParamInternalName Method

Returns the name of the parameter that corresponds to a part on a message of an operation in a WSDL document.

Pascal

```
function GetParamInternalName(Info: PTypeInfo; const MethodName: string; const
ExternalParamName: InvString): string;
```

C++

```
__fastcall UnicodeString GetParamInternalName(PTypeInfo Info, const UnicodeString MethodName,
const InvString ExternalParamName);
```

Description

GetParamInternalName returns the name of the method parameter that corresponds to a specified part. This is the parameter of a method on an invokable interface that is registered to correspond to a part that is defined in a WSDL document. The association between the parameter name and the part name is established by the RegisterExternalParamName (see page 861) method.

Info is the type information pointer for the invokable interface that includes the method.

MethodName is the name of the method on that invokable interface.

ExternalParamName is the name of a part on one of the messages in the operation that corresponds to MethodName. GetParamInternalName returns the corresponding parameter name.

See Also

RegisterExternalParamName (see page 861)

GetParamExternalName (see page 853)

23.1.7.4.21 TInvokableClassRegistry.GetRegInterfaceEntry Method

Returns the internal data that TInvokableClassRegistry (see page 840) maintains for one of its registered interfaces.

Pascal

```
function GetRegInterfaceEntry(Index: Integer): InvRegIntfEntry;
```

C++

```
__fastcall InvRegIntfEntry GetRegInterfaceEntry(int Index);
```

Description

TInvokableClassRegistry (see page 840) maintains an InvRegIntfEntry (see page 937) value for every registered invokable interface. GetRegInterfaceEntry provides indexed access to this information.

The InvRegIntfEntry (see page 937) value for a registered interface is intended for internal use by the invocation registry and

the components that use it.

See Also

GetInfoForURI (see page 848)

GetInterfaceCount (see page 849)

RegisterDefaultSOAPAction (see page 859)

23.1.7.4.22 TInvokableClassRegistry.GetRequestHeaderInfoForInterface Method

Retrieves an array of headers that are registered as associated with requests to an invokable interface.

Pascal

```
function GetRequestHeaderInfoForInterface(Info: PTypeInfo): THeaderItemArray;
```

C++

```
__fastcall THeaderItemArray GetRequestHeaderInfoForInterface(PTypeInfo Info);
```

Description

Typically, applications do not need to call GetRequestHeaderInfoForInterface. This method is used internally by the WSDL publisher to generate the WSDL that describes the headers that are used in messages that invoke a method on an invokable interface.

Info is the runtime type information (RTTI) from a registered invokable interface. The interface must have been previously registered by a call to the RegisterInterface (see page 864) method.

GetRequestHeaderInfoForInterface returns an array of IntfHeaderItem (see page 936) values, each of which describes a header that was registered for request messages on the invokable interface by a call to the RegisterHeaderClass (see page 862) method.

See Also

RegisterHeaderClass (see page 862)

TWSDLHTMLPublish

RegisterInterface (see page 864)

GetHeaderInfoForInterface (see page 846)

23.1.7.4.23 TInvokableClassRegistry.GetResponseHeaderInfoForInterface Method

Retrieves an array of headers that are registered as associated with responses to invocations of an invokable interface.

Pascal

```
function GetResponseHeaderInfoForInterface(Info: PTypeInfo): THeaderItemArray;
```

C++

```
__fastcall THeaderItemArray GetResponseHeaderInfoForInterface(PTypeInfo Info);
```

Description

Typically, applications do not need to call GetResponseHeaderInfoForInterface. This method is used internally by the WSDL publisher to generate the WSDL that describes the headers that are used in response messages that return the results of executing a method on an invokable interface.

Info is the runtime type information (RTTI) from a registered invokable interface. The interface must have been previously registered by a call to the RegisterInterface (see page 864) method.

GetResponseHeaderInfoForInterface returns an array of IntfHeaderItem (see page 936) values, each of which describes a header that was registered for response messages on the invokable interface by a call to the RegisterHeaderClass (see page 862) method.

See Also

RegisterHeaderClass (see page 862)

TWSDLHTMLPublish

RegisterInterface (see page 864)

GetHeaderInfoForInterface (see page 846)

23.1.7.4.24 TInvokableClassRegistry.GetReturnParamNames Method

Returns the list of parameter names that map to return values of interface methods.

Pascal

```
function GetReturnParamNames(const IntfInfo: PTypeInfo): string;
```

C++

```
__fastcall UnicodeString GetReturnParamNames(const PTypeInfo IntfInfo);
```

Description

GetReturnParamNames returns the list of parameter names that have been registered as mapping to return values of methods on a specified interface. Any out messages in a port type with these names are identified as the return value of a method on the corresponding interface. If the application does not explicitly register output parameter names by a call to RegisterReturnParamNames (see page 867), GetReturnParamNames returns the default list, 'Result;Return'.

IntfInfo (see page 789) is the type information pointer for the invokable interface for which you want the registered parameter names.

GetReturnParamNames returns the list of parameter names as a semicolon delimited list.

See Also

RegisterReturnParamNames (see page 867)

23.1.7.4.25 GetUDDIInfo Method

This is the overview for the GetUDDIInfo method overload.

Overload List

	Name	Description
	TInvokableClassRegistry.GetUDDIInfo (PTypeInfo, string, string) (see page 856)	Retrieves the description of the UDDI location that describes a registered interface.
	TInvokableClassRegistry.GetUDDIInfo (TGUID, string, string) (see page 856)	Retrieves the description of the UDDI location that describes a registered interface.

23.1.7.4.25.1 TInvokableClassRegistry.GetUDDIInfo Method (PTypeInfo, string, string)

Retrieves the description of the UDDI location that describes a registered interface.

Pascal

```
function GetUDDIInfo(const IntfInfo: PTypeInfo; var Operator: string; var BindingKey: string):
Boolean; overload;
function GetUDDIInfo(const AGUID: TGUID; var Operator: string; var BindingKey: string):
Boolean; overload;
```

C++

```
__fastcall Boolean GetUDDIInfo(const PTypeInfo IntfInfo, UnicodeString Operator, UnicodeString
BindingKey);
__fastcall Boolean GetUDDIInfo(const TGUID AGUID, UnicodeString Operator, UnicodeString
BindingKey);
```

Description

Typically, applications do not need to call GetUDDIInfo. This method is used internally to provide fail-over support when a registered interface was imported using the UDDI browser. It retrieves the information that identifies the UDDI entry for a Web Service, so that the application can re-fetch the target URL for a registered interface. The UDDI information is previously registered by a call to the RegisterUDDIInfo (see page 867) method.

IntfInfo (see page 789) is the runtime type information (RTTI) from a registered invokable interface. The interface must have been previously registered by a call to the RegisterInterface (see page 864) method.

AGUID is a GUID that identifies the registered interface. The interface must have been previously registered by a call to the RegisterInterface (see page 864) method.

Operator returns the URL of the UDDI registry that was used to import the interface definition.

BindingKey is the unique identifier for the TModel that describes the Web Service to which the interface belongs.

GetUDDIInfo returns true if it successfully locates and returns values for the Operator and BindingKey parameters. It returns false if there is no registered UDDI information for the specified interface.

See Also

RegisterUDDIInfo (see page 867)

RegisterInterface (see page 864)

23.1.7.4.26 GetWSDLEncoding Method

This is the overview for the GetWSDLEncoding method overload.

Overload List

	Name	Description
	TInvokableClassRegistry.GetWSDLEncoding (PTypeInfo) (see page 857)	Returns the encoding attribute of the xml header of a document that defines the port type to which the interface corresponds.
	TInvokableClassRegistry.GetWSDLEncoding (PTypeInfo, string) (see page 857)	Returns the encoding attribute of the xml header of a document that defines the port type to which the interface corresponds.
	TInvokableClassRegistry.GetWSDLEncoding (PTypeInfo, string, string) (see page 857)	Returns the encoding attribute of the xml header of a document that defines the port type to which the interface corresponds.

23.1.7.4.26.1 TInvokableClassRegistry.GetWSDLEncoding Method (PTypeInfo)

Returns the encoding attribute of the xml header of a document that defines the port type to which the interface corresponds.

Pascal

```
function GetWSDLEncoding(Info: PTypeInfo; const Namespace: string; const InternalIntfName:
string): InvString; overload;
function GetWSDLEncoding(Info: PTypeInfo; const Namespace: string): InvString; overload;
function GetWSDLEncoding(Info: PTypeInfo): InvString; overload;
```

C++

```
__fastcall InvString GetWSDLEncoding(PTypeInfo Info, const UnicodeString Namespace, const
UnicodeString InternalIntfName);
__fastcall InvString GetWSDLEncoding(PTypeInfo Info, const UnicodeString Namespace);
__fastcall InvString GetWSDLEncoding(PTypeInfo Info);
```

Description

GetWSDLEncoding returns the encoding attribute, if any, that is registered with the interface by a call to the RegisterInterface (see page 864) method. The encoding attribute is an attribute of the xml header of a WSDL document, and describes the character set that should be used with all the elements that the document defines.

Info points to the type information of the registered interface.

Namespace specifies the namespace that was registered for the interface by the RegisterInterface (see page 864) method. It can be an empty string.

InternalIntfName is the name of the interface (which may differ from the port type to which it corresponds).

See Also

RegisterInterface (see page 864)

23.1.7.4.27 TInvokableClassRegistry.HasRegInterfaceImpl Method

Indicates whether there is a registered implementation class for an invokable interface.

Pascal

```
function HasRegInterfaceImpl(Index: Integer): Boolean;
```

C++

```
__fastcall Boolean HasRegInterfaceImpl(int Index);
```

Description

Typically, applications do not need to call HasRegInterfaceImpl. This method is used internally to determine whether an implementation class has been registered for an invokable interface. The RegisterInvokableClass (see page 865) method registers these implementation classes.

Index identifies the invokable interface for which an implementation class is sought. This index is the same as the index used in the GetRegInterfaceEntry (see page 854) method, and reflects the order in which interfaces are registered.

HasRegInterfaceImpl returns true if there is a registered implementation class for the specified interface, false otherwise.

See Also

GetInterfaceCount (see page 849)

GetRegInterfaceEntry (see page 854)

RegisterInterface (see page 864)

RegisterInvokableClass (see page 865)

23.1.7.4.28 TInvokableClassRegistry.RegisterAllSOAPActions Method

Supplies a set of SOAP Action headers to use for an already-registered interface.

Pascal

```
procedure RegisterAllSOAPActions(Info: PTypeInfo; const AllSOAPActions: InvString);
```

C++

```
__fastcall RegisterAllSOAPActions(PTypeInfo Info, const InvString AllSOAPActions);
```

Description

RegisterDefaultSOAPAction (see page 859) associates a registered invokable interface with a set of SOAP Action headers. A default SOAP Action header is generated (based on the namespace) when you register the interface by a call to the RegisterInterface (see page 864) method. RegisterAllSOAPActions lets you supply or change the SOAP Action header to a set of headers after the interface is already registered.

Info is the runtime type information (RTTI) for the interface for which you are supplying a set of SOAP Action headers.

AllSOAPActions is a string that includes the SOAP headers to use for calls to the interface that Info identifies. The individual header strings are delimited by a character that is given by the first character in the string.

See Also

RegisterInterface (see page 864)

RegisterDefaultSOAPAction (see page 859)

23.1.7.4.29 TInvokableClassRegistry.RegisterDefaultSOAPAction Method

Supplies a SOAP Action header to use for an already-registered interface.

Pascal

```
procedure RegisterDefaultSOAPAction(Info: PTypeInfo; const DefSOAPAction: InvString);
```

C++

```
__fastcall RegisterDefaultSOAPAction(PTypeInfo Info, const InvString DefSOAPAction);
```

Description

RegisterDefaultSOAPAction associates a registered invokable interface with a SOAP Action header. A default SOAP Action header is generated (based on the namespace) when you register the interface by a call to the RegisterInterface (see page 864) method. RegisterDefaultSOAPAction lets you supply or change the SOAP Action header after the interface is already registered.

Info is the runtime type information (RTTI) for the interface for which you are supplying a default SOAP Action header.

DefSOAPAction is the SOAP Action header to use for calls to the interface that Info identifies. If this is an empty string, the server automatically generates a SOAP Action header based on the namespace in which the corresponding port type is defined.

See Also

RegisterInterface (see page 864)

THTTPRIO

RegisterAllSOAPActions (see page 859)

23.1.7.4.30 TInvokableClassRegistry.RegisterException Method

Registers the remotable exception classes that calls to an invokable interface can raise.

Pascal

```
procedure RegisterException(Info: PTypeInfo; AClass: TClass);
```

C++

```
__fastcall RegisterException(PTypeInfo Info, TClass AClass);
```

Description

Use RegisterException to register a remotable exception class. When an exception class is registered, its definition is exported to WSDL documents so that it can be used to communicate exception information between clients and servers.

Info points to the type information of the invokable interface that can raise the exception. If more than one interface can raise the exception, you must call this method once for each interface type.

AClass describes the exception class. This must be the class reference for an ERemotableException (see page 810) descendant.

Note: After a call to RegisterException, the exception class is registered so that it is associated with all methods of the specified interface. If only a subset of the interface's methods raise the exception, you can indicate this by calling the RegisterExceptionMethod (see page 860) method for each method that can raise the exception.

See Also

RegisterInterface (see page 864)

ERemotableException (see page 810)

RegisterExceptionMethod (see page 860)

Creating Custom Exception Classes for Web Services

23.1.7.4.31 TInvokableClassRegistry.RegisterExceptionMethod Method

Registers methods of an invokable interface that can raise a registered exception class.

Pascal

```
procedure RegisterExceptionMethod(Info: PTypeInfo; AClass: TClass; const MethodName: string);
```

C++

```
__fastcall RegisterExceptionMethod(PTypeInfo Info, TClass AClass, const UnicodeString
MethodName);
```

Description

Call RegisterExceptionMethod for each method of an invokable interface that can raise a registered exception class. The exception class must be previously registered by a call to the RegisterException (see page 860) method. If you do not call RegisterExceptionMethod, then the WSDL document for the interface treats the exception as associated with all methods of the interface. If you want to limit the use of the exception to a subset of the methods, call RegisterExceptionMethod for each method that can raise the exception. After the first method is registered this way, the exception becomes limited to that method alone (and to any method subsequently registered by another call to RegisterExceptionMethod).

Info points to the type information of the invokable interface that can raise the exception.

AClass describes the exception class. This must be the class reference used to register the exception previously by a call to the

RegisterException (see page 860) method.

MethodName is the name of a method that can raise the exception.

See Also

RegisterInterface (see page 864)

ERemotableException (see page 810)

RegisterException (see page 860)

23.1.7.4.32 TInvokableClassRegistry.RegisterExternalMethName Method

Registers the correspondence between a method name on an invokable interface and the corresponding operation name in a WSDL document.

Pascal

```
procedure RegisterExternalMethName(Info: PTypeInfo; const InternalName: string; const
ExternalName: InvString);
```

C++

```
__fastcall RegisterExternalMethName(PTypeInfo Info, const UnicodeString InternalName, const
InvString ExternalName);
```

Description

Use RegisterExternalMethName to create an association between the name of an operation in a WSDL document and the corresponding method of the invokable interface that corresponds to that operation's port type. This allows the invokable interface to represent a port type that includes operations with names that are not legal in Object Pascal (for example, keywords).

Info points to the type information of the invokable interface.

InternalName is the name used for the method of the invokable interface that corresponds to the operation.

ExternalName is the name of the operation that is defined in the WSDL document.

See Also

RegisterExternalParamName (see page 861)

RegisterInterface (see page 864)

23.1.7.4.33 TInvokableClassRegistry.RegisterExternalParamName Method

Registers the correspondence between a parameter name on an invokable interface and the corresponding part name in a WSDL document.

Pascal

```
procedure RegisterExternalParamName(Info: PTypeInfo; const MethodName: string; const
InternalName: string; const ExternalName: InvString);
```

C++

```
__fastcall RegisterExternalParamName(PTypeInfo Info, const UnicodeString MethodName, const
UnicodeString InternalName, const InvString ExternalName);
```

Description

Use RegisterExternalParamName to create an association between the name of a part in a WSDL document and the corresponding parameter on a method of an invokable interface. This allows the invokable interface to handle definitions that have names that are not legal in Object Pascal (for example, keywords).

Info points to the type information of the invokable interface.

MethodName is the name of the method on the invokable interface.

InternalName is the name used for the parameter of the specified method that corresponds to the part with the name specified by ExternalName.

ExternalName is the name of the part that is defined in the WSDL document.

See Also

RegisterExternalMethName (see page 861)

RegisterInterface (see page 864)

23.1.7.4.34 RegisterHeaderClass Method

This is the overview for the RegisterHeaderClass method overload.

Overload List

	Name	Description
	TInvokableClassRegistry.RegisterHeaderClass (PTypeInfo, TClass, eHeaderMethodType, Boolean) (see page 862)	Registers a class that represents a headers which accompanies calls to an invokable interface.
	TInvokableClassRegistry.RegisterHeaderClass (PTypeInfo, TClass, InvString, InvString, eHeaderMethodType, Boolean) (see page 862)	Registers a class that represents a headers which accompanies calls to an invokable interface.

23.1.7.4.34.1 TInvokableClassRegistry.RegisterHeaderClass Method (PTypeInfo, TClass, InvString, InvString, eHeaderMethodType, Boolean)

Registers a class that represents a headers which accompanies calls to an invokable interface.

Pascal

```
procedure RegisterHeaderClass(Info: PTypeInfo; AClass: TClass; const HeaderName: InvString;
const HeaderNamespace: InvString; DefaultMethodType: eHeaderMethodType = hmtAll; Required:
Boolean = False); overload;
procedure RegisterHeaderClass(Info: PTypeInfo; AClass: TClass; DefaultMethodType:
eHeaderMethodType = hmtAll; Required: Boolean = False); overload;
```

C++

```
__fastcall RegisterHeaderClass(PTypeInfo Info, TClass AClass, const InvString HeaderName,
const InvString HeaderNamespace, eHeaderMethodType DefaultMethodType = hmtAll, Boolean
Required = False);
__fastcall RegisterHeaderClass(PTypeInfo Info, TClass AClass, eHeaderMethodType
DefaultMethodType = hmtAll, Boolean Required = False);
```

Description

Use RegisterHeaderClass to register a TSOAPHeader (see page 908) descendant. Registering a TSOAPHeader (see page 908) descendant serves two purposes:

On servers, when a TSOAPHeader (see page 908) descendant is registered, its definition is exported to WSDL documents so that applications using that WSDL document can recognize and process the header.

On both servers and clients, registering associates the TSOAPHeader (see page 908) descendant with a header name and the namespace in which that name is defined.

Info points to the type information of the invokable interface that the header should accompany. If the header should be included

in messages invoking more than one interface or in the responses to more than one interface, you must call this method once for each interface type.

AClass describes the TSOAPHeader (see page 908) descendant that represents the header.

HeaderName is the name of the header, as defined within the header's namespace. If HeaderName is not supplied, the header's name is assumed to be the same as the type name under which it is registered in the remotable type registry.

HeaderNamespace is the URI for the namespace in which the header's name is defined. If HeaderName and HeaderNamespace are not supplied, the header is assumed to be defined in the same namespace as the interface with which it is associated.

DefaultMethodType indicates whether the header is to be used on request messages, response messages, or both. This default value is associated with the interface as a whole. It can be changed for individual methods of the interface by subsequent calls to the RegisterHeaderMethod (see page 863) method.

Required indicates whether the header must be included in messages about the interface. When Required is true, the header must be included and understood by the recipient of the message. When Required is false, the header is optional. Note that if the header is required by some but not all messages, you must set this Required parameter to false, and then add the requirement to individual methods by subsequent calls to the RegisterHeaderMethod (see page 863) method.

Note: After a call to RegisterHeaderClass, the header class is registered so that it is associated with all methods of the specified interface. If only a subset of the interface's methods use the header, or if the header should have a different method type for some of the methods on the interface, or if the header is required by a subset of the methods on the interface, you can indicate this by calling the RegisterHeaderMethod (see page 863) method for each method that the header accompanies.

See Also

RegisterInterface (see page 864)

TSOAPHeader (see page 908)

RegisterHeaderMethod (see page 863)

23.1.7.4.35 TInvokableClassRegistry.RegisterHeaderMethod Method

Registers methods of an invokable interface that make use of a registered header.

Pascal

```
procedure RegisterHeaderMethod(Info: PTypeInfo; AClass: TClass; const MethodName: string;
MethodType: eHeaderMethodType = hmtAll; Required: Boolean = False); overload;
```

C++

```
__fastcall RegisterHeaderMethod(PTypeInfo Info, TClass AClass, const UnicodeString MethodName,
eHeaderMethodType MethodType = hmtAll, Boolean Required = False);
```

Description

Call RegisterHeaderMethod for each method of an invokable interface that uses a registered header type. The header class must be previously registered by a call to the RegisterHeaderClass (see page 862) method. If you do not call RegisterHeaderMethod, then the WSDL document for the interface treats the header as associated with all methods of the interface. Use RegisterHeaderMethod if you want to

Limit the use of the header to a subset of the methods.

Specify whether the header is used for request messages, response messages, or both in a way that varies from method to method.

Indicate that the header is required for some methods but not others.

Call RegisterHeaderMethod for each method that uses the header. After the first method is registered this way, the header becomes limited to that method alone (and to any method subsequently registered by another call to RegisterHeaderMethod).

Info points to the type information of the invokable interface that the header should accompany.

AClass describes the TSOAPHeader (see page 908) descendant that represents the header. This class must be previously registered by a call to RegisterHeaderClass (see page 862).

MethodName is the name of a method that uses the header.

MethodType indicates whether the header is to be used on request messages, response messages, or both.

Required indicates whether the header must be included in messages about the specified method. When Required is true, the header must be included and understood by the recipient of the message. When Required is false, the header is optional.

See Also

RegisterInterface (see page 864)

TSOAPHeader (see page 908)

RegisterHeaderClass (see page 862)

23.1.7.4.36 TInvokableClassRegistry.RegisterInterface Method

Registers an invokable interface.

Pascal

```
procedure RegisterInterface(Info: PTypeInfo; const Namespace: InvString = ''; const
WSDLEncoding: InvString = ''; const Doc: string = ''; const ExtName: InvString = '');
```

C++

```
__fastcall RegisterInterface(PTypeInfo Info, const InvString Namespace = '', const InvString
WSDLEncoding = '', const UnicodeString Doc = '', const InvString ExtName = '');
```

Description

Call RegisterInterface to register an invokable interface with the invocation registry. Typically, the call to RegisterInterface is added by the WSDL importer or Web Service wizard. In Delphi, this call goes in the initialization section of the unit that defines the invokable interface. In C++, it is called within a RegTypes method that uses the pragma startup directive.

Once the invokable interface is registered, client applications can statically bind to it using THTTPRio. On the server side, you must also register the implementation class (using the RegisterInvokableClass (see page 865) method) before the server can respond to incoming SOAP requests.

PInfo is a pointer to the type information for the interface. You can obtain the type information from an invokable interface by calling the TypeInfo function (Delphi) or __delphirtti (C++).

Namespace specifies a namespace URI that uniquely identifies the interface. If Namespace is set to an empty string, RegisterInterface automatically generates a unique string to use as a namespace URI. Note that RegisterInterface generates a SOAP Action header to use based on the Namespace of the interface. You can change this SOAP Action header using the RegisterDefaultSOAPAction (see page 859) method.

WSDLEncoding specifies a encoding attribute that appears on the xml header of any WSDL document that includes the definition of this interface (or rather, the port type to which it corresponds). The encoding attribute describes the character set to use in the WSDL document.

Doc is a descriptive string that describes the purpose or use of the interface. It is used if the application publishes a WSDL document that defines this interface.

ExtName is the name of the port type that corresponds to this interface in a WSDL document. This can be an empty string if the

port type and interface have the same name. It is useful when a port type in a WSDL document has a name that is not a valid Object Pascal identifier (for example, a keyword).

DefSOAPAction is the SOAP Action header that should be used for messages that represent calls to the methods on this interface. If you do not supply a default SOAP Action header, your application uses an automatically-generated header that is based on the namespace of the interface.

See Also

RegisterInvokableClass (see page 865)

IInvokable

THTTPRIO

UnRegisterInterface (see page 867)

RegisterDefaultSOAPAction (see page 859)

23.1.7.4.37 RegisterInvokableClass Method

This is the overview for the RegisterInvokableClass method overload.

Overload List

	Name	Description
	TInvokableClassRegistry.RegisterInvokableClass (TClass) (see page 865)	Registers the implementation class for an invokable interface.
	TInvokableClassRegistry.RegisterInvokableClass (TClass, TCreateInstanceProc) (see page 865)	Registers the implementation class for an invokable interface.

23.1.7.4.37.1 TInvokableClassRegistry.RegisterInvokableClass Method (TClass)

Registers the implementation class for an invokable interface.

Pascal

```
procedure RegisterInvokableClass(AClass: TClass; CreateProc: TCreateInstanceProc); overload;
procedure RegisterInvokableClass(AClass: TClass); overload;
```

C++

```
__fastcall RegisterInvokableClass(TClass AClass, TCreateInstanceProc CreateProc);
__fastcall RegisterInvokableClass(TClass AClass);
```

Description

RegisterInvokableClass registers the class that implements an already-registered invokable interface. It is only needed on server applications that implement Web Services. The interface itself is registered by the RegisterInterface (see page 864) method.

The invocation registry must be able to instantiate the implementation class. If the class has a virtual constructor that takes no arguments, you do not need to supply a method to create the class instance. If the constructor takes any arguments, of if it is not virtual, you must supply a method to instantiate the class as the value of the CreateProc argument.

AClass is a class reference to the class being registered. In Delphi code, you need only supply the class name. In C++, you can obtain a class reference for a class using the __classid operator.

CreateProc is a procedure that returns an instance of AClass. If CreateProc is nil (Delphi) or NULL (C++), then AClass must be a descendant of TInvokableClass (see page 833).

See Also

RegisterInterface (see page 864)

UnRegisterInvokableClass (see page 868)

TInvokableClass (see page 833)

23.1.7.4.38 RegisterInvokeOptions Method

This is the overview for the RegisterInvokeOptions method overload.

Overload List

	Name	Description
	TInvokableClassRegistry.RegisterInvokeOptions (PTypeInfo, TIntfInvokeOption) (see page 866)	Registers flags that customize the way calls to a registered interface are made.
	TInvokableClassRegistry.RegisterInvokeOptions (PTypeInfo, TIntfInvokeOptions) (see page 866)	Registers flags that customize the way calls to a registered interface are made.

23.1.7.4.38.1 TInvokableClassRegistry.RegisterInvokeOptions Method (PTypeInfo, TIntfInvokeOption)

Registers flags that customize the way calls to a registered interface are made.

Pascal

```
procedure RegisterInvokeOptions(Info: PTypeInfo; const InvokeOption: TIntfInvokeOption);
overload;
procedure RegisterInvokeOptions(Info: PTypeInfo; const InvokeOptions: TIntfInvokeOptions);
overload;
```

C++

```
__fastcall RegisterInvokeOptions(PTypeInfo Info, const TIntfInvokeOption InvokeOption);
__fastcall RegisterInvokeOptions(PTypeInfo Info, const TIntfInvokeOptions InvokeOptions);
```

Description

RegisterInvokeOptions associates the specified invocation flag or flags with a registered interface. The interface must be previously registered by a call to the RegisterInterface (see page 864) method. Invocation flags customize the way a registered interface is called, especially the way such calls are marshaled or unmarshaled.

Info identifies the registered interface using its runtime type information (RTTI). You can obtain the type information from an invokable interface by calling the TypeInfo function (delphi) or __delphirtti (C++).

InvokeOption specifies a single invocation option to associate with the interface.

InvokeOptions specifies a set of invocation options to associate with the interface.

Note: RegisterInvokeOptions adds the specified option or options to the set of invocation flags already registered for the interface. It does not remove any previously registered options.

See Also

GetIntfInvokeOptions (see page 851)

Converter

Options (see page 2781)

RegisterInterface (see page 864)

23.1.7.4.39 TInvokableClassRegistry.RegisterReturnParamNames Method

Registers a list of parameter names that map to return values of interface methods.

Pascal

```
procedure RegisterReturnParamNames(Info: PTypeInfo; const RetParamNames: string);
```

C++

```
__fastcall RegisterReturnParamNames(PTypeInfo Info, const UnicodeString RetParamNames);
```

Description

RegisterReturnParamNames registers a list of parameter names that correspond to the return value of methods on a specified interface. Typically, this method is called by the WSDL importer. It is important that these names correspond exactly to those that the WSDL importer uses if the interface was imported from a WSDL document.

Info is the type information pointer for the invokable interface whose method return values map to the names in the list.

RetParamNames is a semicolon delimited list of parameter names that map to return values of methods on the specified interface. If RetParamNames is an empty string, the methods of the interface never have a return value (only output parameters). If no return parameter names are registered for an interface, the default list, 'Result;Return', is used.

See Also

GetReturnParamNames (see page 856)

23.1.7.4.40 TInvokableClassRegistry.RegisterUDDIInfo Method

Registers the location of the UDDI description that can be used to import the location of a Web Service.

Pascal

```
procedure RegisterUDDIInfo(Info: PTypeInfo; const Operator: String; const BindingKey: string);
```

C++

```
__fastcall RegisterUDDIInfo(PTypeInfo Info, const UnicodeString Operator, const UnicodeString
BindingKey);
```

Description

Typically, applications do not need to call RegisterUDDIInfo. This method is called internally when a registered interface is imported using the UDDI browser. It associates the location of the imported UDDI entry with a registered interface so that the application can later provide fail-over support by re-fetching the location of the Web Service.

Info is the runtime type information (RTTI) from a registered invokable interface. The interface must have been previously registered by a call to the RegisterInterface (see page 864) method.

Operator is the URL of the UDDI registry that was used to import the interface definition.

BindingKey is the unique identifier for the TModel that describes the Web Service to which the interface belongs.

See Also

GetUDDIInfo (see page 856)

RegisterInterface (see page 864)

23.1.7.4.41 TInvokableClassRegistry.UnRegisterInterface Method

Unregisters an invokable interface.

Pascal

```
procedure UnRegisterInterface(Info: PTypeInfo);
```

C++

```
__fastcall UnRegisterInterface(PTypeInfo Info);
```

Description

Call UnRegisterInterface to unregister an invokable interface from the invocation registry. In Delphi, this call goes in the finalization section of the unit that defines the invokable interface. In C++, it is called from a method that uses the pragma exit directive.

Once an invokable interface is unregistered, client applications can no longer call it and server applications can't respond to calls directed at the interface methods.

Info is a pointer to the runtime type information for the interface. You can obtain the type information from an invokable interface by calling the TypeInfo function in Delphi or __delphirtti in C++.

See Also

UnRegisterInvokableClass (see page 868)

IInvokable

RegisterInterface (see page 864)

23.1.7.4.42 TInvokableClassRegistry.UnRegisterInvokableClass Method

Unregisters the implementation class for an invokable interface.

Pascal

```
procedure UnRegisterInvokableClass(AClass: TClass);
```

C++

```
__fastcall UnRegisterInvokableClass(TClass AClass);
```

Description

UnRegisterInvokableClass unregisters an implementation class that was previously registered by a call to RegisterInvokableClass (see page 865). Once a class is unregistered, the invocation registry can no longer create and use instances of the specified class to handle incoming method calls to its corresponding invokable interface.

AClass is a class reference to the class being unregistered.

See Also

UnRegisterInterface (see page 867)

RegisterInvokableClass (see page 865)

23.1.8 TRemotable Class

TRemotable is the base class for classes that can be passed as parameters or return values in a Web Service application.

Class Hierarchy

Pascal

```
TRemotable = class;
```

C++

```
class TRemotable;
```

File

InvokeRegistry

Description

Use TRemotable as a base class when defining a class that can be used for parameters or return values on an invokable interface. TRemotable descendants can represent non-scalar types, or scalar types that do not correspond to Object Pascal types. TRemotable is compiled with runtime type information (RTTI) and has a virtual constructor that the remotable class registry recognizes and uses to supply class instances.

Note: To create a class that represents scalar values that do not correspond directly to an Object Pascal type, use TRemotableXS (see page 892) as a base class instead. TRemotableXS (see page 892) is a descendant of the TRemotable class.

Note: When writing a TRemotable descendant, you can control whether a property is encoded as an element node or as an attribute node by using the stored directive. Properties declared with a stored directive set to AS_ATTRIBUTE (see page 956) are encoded as attribute nodes. Properties that are declared without a stored directive, or with a stored directive set to anything other than AS_ATTRIBUTE (see page 956) are encoded as element nodes.

Both the client application and server application must register a TRemotable descendant with the remotable class registry before they can use it. To register a remotable class, use the RegisterXSClass method of the object that the global RemTypeRegistry (see page 931) function returns.

On server applications, instances of TRemotable descendants that are input parameters on an invokable interface are automatically created when the method call is unmarshaled and automatically freed after any output parameters or return value are marshaled so that they can be sent to the client. When a TRemotable descendant is created in a method that was called remotely using an invokable interface, the instance is automatically freed after the value of the TRemotable descendant is marshaled for transport back to the client application.

The caller of an invokable interface (client) is responsible for creating any TRemotable instances that it passes as input parameters, and for freeing any TRemotable instances it creates or that are returned as output parameters or method results.

See Also

RemClassRegistry (see page 930)

RemTypeRegistry (see page 931)

TRemotableTypeRegistry (see page 874)

IInvokable

Using Remotable Objects

23.1.8.1 TRemotable Members

The following tables list the members exposed by TRemotable.

Protected Fields

	Name	Description
	FSerializationOptions (see page 871)	This is FSerializationOptions, a member of class TRemotable.

Public Methods

	Name	Description
	Create (see page 870)	Creates and initializes a TRemotable (see page 868) object.
	Destroy (see page 870)	Frees the memory associated with the TRemotable (see page 868) object.
	ObjectToSOAP (see page 871)	Converts this remotable object to its SOAP representation.
	SOAPToObject (see page 872)	Initializes this TRemotable (see page 868) instance to represent the value of its SOAP representation.

Public Properties

	Name	Description
	DataContext (see page 873)	Specifies the object that controls the lifetime of a TRemotable (see page 868) instance.
	SerializationOptions (see page 873)	Indicates how the value of the TRemotable (see page 868) descendant encodes its value.

23.1.8.2 TRemotable.Create Constructor

Creates and initializes a TRemotable (see page 868) object.

Pascal

```
constructor Create; virtual;
```

C++

```
virtual __fastcall TRemotable();
```

Description

The TRemotable (see page 868) constructor does nothing special. It exists primarily because

The remotable type registry knows how to call it.

It is virtual, so that the remotable type registry can also instantiate descendant classes.

See Also

RemTypeRegistry (see page 931)

23.1.8.3 TRemotable.Destroy Destructor

Frees the memory associated with the TRemotable (see page 868) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRemotable();
```

Description

Use the Free method to free a TRemotable (see page 868) instance. Free checks that the object is not nil before calling Destroy.

On server applications, there is typically no need to explicitly free a TRemotable (see page 868) instance (for example, one

created as a return value or output parameter). By default, when a TRemotable (see page 868) descendant is created in a method that was called remotely using an invokable interface, it is added to a data context (the value of the DataContext (see page 873) property). As long as the remotable object belongs to the data context, the data context handles freeing the object. Similarly, TRemotable (see page 868) instances that are passed in as a parameter belong to the data context.

On client applications, TRemotable (see page 868) instances created and passed to an invokable interface or returned as a parameter or result of an invokable interface must be freed by the calling application.

See Also

System::TObject::Free

DataContext (see page 873)

23.1.8.4 TRemotable Fields

The fields of the TRemotable class are listed here.

Protected Fields

	Name	Description
	FSerializationOptions (see page 871)	This is FSerializationOptions, a member of class TRemotable.

23.1.8.4.1 TRemotable.FSerializationOptions Field

Pascal

```
FSerializationOptions: TSerializationOptions;
```

C++

```
TSerializationOptions FSerializationOptions;
```

Description

This is FSerializationOptions, a member of class TRemotable.

23.1.8.5 TRemotable Methods

The methods of the TRemotable class are listed here.

Public Methods

	Name	Description
	ObjectToSOAP (see page 871)	Converts this remotable object to its SOAP representation.
	SOAPToObject (see page 872)	Initializes this TRemotable (see page 868) instance to represent the value of its SOAP representation.

23.1.8.5.1 TRemotable.ObjectToSOAP Method

Converts this remotable object to its SOAP representation.

Pascal

```
function ObjectToSOAP(RootNode: IXMLNode; ParentNode: IXMLNode; const ObjConverter:
IObjConverter; const Name: InvString; const URI: InvString; ObjConvOpts:
TObjectConvertOptions; out RefID: InvString): IXMLNode; virtual;
```

C++

```
virtual __fastcall IXMLNode ObjectToSOAP(IXMLNode RootNode, IXMLNode ParentNode, const
IObjConverter ObjConverter, const InvString Name, const InvString URI, TObjectConvertOptions
ObjConvOpts, InvString RefID);
```

Description

Override ObjectToSOAP to customize the way your application converts a TRemotable (see page 868) descendant to its SOAP representation. In TRemotable (see page 868), ObjectToSOAP simply calls the ObjInstanceToSOAP method of the converter that is supplied by the ObjConverter parameter.

RootNode is an ancestor of the generated node to which information about the generated node can be added as attributes.

ParentNode is the node in the SOAP representation that should act as a parent to the node that ObjectToSOAP generates.

ObjConverter is the interface of the default converter that translates between remotable objects and their SOAP representation. This is the interface to a TOPToSoapDomConvert instance.

Name is the name of the node that is generated.

URI is the URI for the namespace in which Name is defined.

ObjConvOpts are flags that customize the way the node is serialized.

RefID returns a reference ID for the node. This is used if the node is multi-referenced.

ObjectToSOAP returns the XML for the newly generated node.

See Also

ObjInstanceToSOAP (see page 922)

SOAPToObject (see page 872)

23.1.8.5.2 TRemotable.SOAPToObject Method

Initializes this TRemotable (see page 868) instance to represent the value of its SOAP representation.

Pascal

```
procedure SOAPToObject(const RootNode: IXMLNode; const Node: IXMLNode; const ObjConverter:
IObjConverter); virtual;
```

C++

```
virtual __fastcall SOAPToObject(const IXMLNode RootNode, const IXMLNode Node, const
IObjConverter ObjConverter);
```

Description

SOAPToObject initializes the published properties of this object instance so that they reflect the header's SOAP representation. Override this method to customize the way your application creates a specific remotable type from its XML representation.

RootNode is an ancestor node of the node for this object that can contain information about the remotable type's value.

Node is the node in the SOAP envelope that represents the remotable type's value.

ObjConverter is the interface of the default converter that translates between remotable objects and their SOAP representation. This is the interface to a TOPToSoapDomConvert instance.

In TRemotable (see page 868), SOAPToObject simply calls the InitObjectFromSOAP method of the converter supplied by the ObjConverter parameter.

See Also

ObjectToSOAP (see page 871)

InitObjectFromSOAP (see page 922)

23.1.8.6 TRemotable Properties

The properties of the TRemotable class are listed here.

Public Properties

	Name	Description
	DataContext (see page 873)	Specifies the object that controls the lifetime of a TRemotable (see page 868) instance.
R	SerializationOptions (see page 873)	Indicates how the value of the TRemotable (see page 868) descendant encodes its value.

23.1.8.6.1 TRemotable.DataContext Property

Specifies the object that controls the lifetime of a TRemotable (see page 868) instance.

Pascal

```
property DataContext: TDataContext;
```

C++

```
__property TDataContext DataContext;
```

Description

By default, when a Web Service application creates an instance of a remotable object, that object exists only for the time it takes to marshal the information it represents or to unmarshal that information into the TRemotable (see page 868) instance and then make a method call that uses the instance. The object specified by DataContext handles the freeing of the TRemotable (see page 868) instance.

There are times when a server may not want the TRemotable (see page 868) instance to be so transient. For example, if the object contains state information, it may be more efficient to have a single instance that is used for every message call. If you change DataContext, the TRemotable (see page 868) instance is removed from the data context, and is not freed when the data context goes away at the end of the method call.

Note: If DataContext is nil (Delphi) or NULL (C++), the application is responsible for freeing the TRemotable (see page 868) instance. For example, TRemotable (see page 868) objects created on client applications are created with a nil (Delphi) or NULL (C++) value for DataContext and must be freed by the client application.

See Also

Using Remotable Objects

23.1.8.6.2 TRemotable.SerializationOptions Property

Indicates how the value of the TRemotable (see page 868) descendant encodes its value.

Pascal

```
property SerializationOptions: TSerializationOptions;
```

C++

```
__property TSerializationOptions SerializationOptions;
```

Description

SerializationOptions is used internally when an instance of TRemotable (see page 868) is converted to or from its SOAP encoding. The value of this property is a set of flags that configure the mapping between the published properties of the TRemotable (see page 868) descendant and the corresponding SOAP encoding.

Typically, the value of this read-only property is set by the constructor of a TRemotable (see page 868) descendant or by the WSDL importer when it registers a remotable class.

See Also

SerializeOptions (see page 887)

RegisterSerializeOptions (see page 884)

23.1.9 TRemotableTypeRegistry Class

TRemotableTypeRegistry is the type of the remotable type registry.

Class Hierarchy

Pascal

```
TRemotableTypeRegistry = class;
```

C++

```
class TRemotableTypeRegistry;
```

File

InvokeRegistry

Description

TRemotableTypeRegistry manages information about remotable classes and types that can be communicated between client applications and a Web Service provider. Remotable classes can be used to represent non-scalar types for parameters and return values, or they can represent exceptions that occur when the Web Service application is trying to execute a request. All remotable classes are compiled with runtime type information (RTTI) and have a virtual constructor that the remotable type registry recognizes and uses to supply class instances. Remotable types include dynamic arrays, enumerated types, and boolean types, which can be used in an invokable interface without resorting to a remotable class representation, but which must be registered because their values require special marshaling.

Both the client application and server application must register remotable classes and types with the remotable type registry before they can use them. Use the RegisterXSClass (see page 885) method to register a remotable class. Use the RegiterXSInfo method to register a dynamic array type or enumerated type.

Do not create instances of TRemotableTypeRegistry. Instead, use the global RemTypeRegistry (see page 931) function to access the remotable type registry. Most applications need only call the RegisterXSClass (see page 885) or RegisterXSInfo (see page 886) methods. The other TRemotableTypeRegistry methods are used internally to create or look up information about remotable classes and types.

See Also

RemTypeRegistry (see page 931)

TRemotable (see page 868)

IInvokable

ERemotableException (see page 810)

TInvokableClassRegistry (see page 840)

23.1.9.1 TRemotableTypeRegistry Members

The following tables list the members exposed by TRemotableTypeRegistry.

Protected Methods

	Name	Description
	Lock (see page 883)	Prevents other threads from accessing the type registry.
	UnLock (see page 888)	Allows other threads to access the type registry after it was locked.

Public Methods

	Name	Description
	ClassOptions (see page 877)	Indicates how to handle multiple references to the same object when converting a specified remotable class to its SOAP representation.
	ClassToURI (see page 878)	This is the overview for the ClassToURI method overload.
	Create (see page 876)	Creates and initializes a TRemotableTypeRegistry (see page 874) object.
	Destroy (see page 876)	Frees the memory associated with the TRemotableTypeRegistry (see page 874) object.
	GetExternalPropName (see page 879)	Returns the name of the XML element that corresponds to a property of a registered remotable class.
	GetInternalPropName (see page 879)	Returns the name of the property on a remotable class that corresponds to a specified XML element.
	GetURICount (see page 879)	Returns the number of entries in the remotable type registry.
	GetURIMap (see page 880)	This is the overview for the GetURIMap method overload.
	GetVarTypeFromXSD (see page 881)	Returns the variant type that corresponds to a registered remotable type.
	GetXSDInfoForClass (see page 881)	Returns type name and namespace URI that correspond to a remotable class.
	InfoToURI (see page 882)	Returns the namespace URI and type name for a registered type.
	IsClassScalar (see page 882)	Indicates whether a registered class represents a scalar type.
	RegisterExternalPropName (see page 883)	Registers the correspondence between a property name on a remotable class and the corresponding element name in an XML schema.
	RegisterHolderClsMember (see page 883)	Registers a remotable type whose type information must be extracted from a holder class.
	RegisterSerializeOptions (see page 884)	This is the overview for the RegisterSerializeOptions method overload.
	RegisterXSClass (see page 885)	Registers a remotable class.
	RegisterXSInfo (see page 886)	Registers a remotable type.
	SerializeOptions (see page 887)	This is the overview for the SerializeOptions method overload.
	TypeInfoToXSD (see page 887)	Returns the type name and namespace URI registered with a remotable class or type.
	UnRegisterXSClass (see page 888)	Unregisters a remotable class.
	UnRegisterXSInfo (see page 889)	Unregisters a remotable type.
	URIToClass (see page 889)	This is the overview for the URIToClass method overload.
	URIToInfo (see page 890)	Returns the type information pointer for a registered remotable class or type.
	VariantToInfo (see page 890)	Returns the TypeInfo pointer for the type of a specified Variant's data.
	XSDToTypeInfo (see page 891)	Returns the type information pointer associated with a specified type name.

Public Properties

	Name	Description
	AutoRegisterNativeTypes (see page 891)	Specifies whether types that are used in an invokable interface or by a remotable class are automatically registered.

23.1.9.2 TRemotableTypeRegistry.Create Constructor

Creates and initializes a TRemotableTypeRegistry (see page 874) object.

Pascal

```
constructor Create;
```

C++

```
__fastcall TRemotableTypeRegistry();
```

Description

Do not create your own instances of TRemotableTypeRegistry (see page 874). Instead, use the global RemTypeRegistry (see page 931) function, which provides access to a single global instance of TRemotableTypeRegistry (see page 874).

See Also

RemTypeRegistry (see page 931)

23.1.9.3 TRemotableTypeRegistry.Destroy Destructor

Frees the memory associated with the TRemotableTypeRegistry (see page 874) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRemotableTypeRegistry();
```

Description

Do not destroy the global TRemotableTypeRegistry (see page 874) instance that is available through the RemClassRegistry (see page 930) function. This object is freed automatically when the application unloads the InvokeRegistry (see page 807) unit.

See Also

RemClassRegistry (see page 930)

23.1.9.4 TRemotableTypeRegistry Methods

The methods of the TRemotableTypeRegistry class are listed here.

Protected Methods

	Name	Description
	Lock (see page 883)	Prevents other threads from accessing the type registry.
	UnLock (see page 888)	Allows other threads to access the type registry after it was locked.

Public Methods

	Name	Description
	ClassOptions (see page 877)	Indicates how to handle multiple references to the same object when converting a specified remotable class to its SOAP representation.
	ClassToURI (see page 878)	This is the overview for the ClassToURI method overload.
	GetExternalPropName (see page 879)	Returns the name of the XML element that corresponds to a property of a registered remotable class.

	GetInternalPropName (see page 879)	Returns the name of the property on a remotable class that corresponds to a specified XML element.
	GetURICount (see page 879)	Returns the number of entries in the remotable type registry.
	GetURIMap (see page 880)	This is the overview for the GetURIMap method overload.
	GetVarTypeFromXSD (see page 881)	Returns the variant type that corresponds to a registered remotable type.
	GetXSDInfoForClass (see page 881)	Returns type name and namespace URI that correspond to a remotable class.
	InfoToURI (see page 882)	Returns the namespace URI and type name for a registered type.
	IsClassScalar (see page 882)	Indicates whether a registered class represents a scalar type.
	RegisterExternalPropName (see page 883)	Registers the correspondence between a property name on a remotable class and the corresponding element name in an XML schema.
	RegisterHolderClsMember (see page 883)	Registers a remotable type whose type information must be extracted from a holder class.
	RegisterSerializeOptions (see page 884)	This is the overview for the RegisterSerializeOptions method overload.
	RegisterXSClass (see page 885)	Registers a remotable class.
	RegisterXSInfo (see page 886)	Registers a remotable type.
	SerializeOptions (see page 887)	This is the overview for the SerializeOptions method overload.
	TypeInfoToXSD (see page 887)	Returns the type name and namespace URI registered with a remotable class or type.
	UnRegisterXSClass (see page 888)	Unregisters a remotable class.
	UnRegisterXSInfo (see page 889)	Unregisters a remotable type.
	URIToClass (see page 889)	This is the overview for the URIToClass method overload.
	URIToInfo (see page 890)	Returns the type information pointer for a registered remotable class or type.
	VariantToInfo (see page 890)	Returns the TypeInfo pointer for the type of a specified Variant's data.
	XSDToTypeInfo (see page 891)	Returns the type information pointer associated with a specified type name.

23.1.9.4.1 TRemotableTypeRegistry.ClassOptions Method

Indicates how to handle multiple references to the same object when converting a specified remotable class to its SOAP representation.

Pascal

```
function ClassOptions(AClass: TClass): TObjMultiOptions;
```

C++

```
__fastcall TObjMultiOptions ClassOptions(TClass AClass);
```

Description

When an application converts a remotable class to its SOAP representation, it can convert multiple references to the same object in one of two ways. It can treat the object as if its properties have meaning, but not the reference itself. (That is, two references to the same object are equivalent values, but not the same thing.) Alternately, if the object reference itself has meaning, it can encode the fact that these are multiple references to the same entity in the SOAP packet.

Individual remotable classes can specify how they want this conversion to be handled, overriding any defaults that would otherwise be used. For example, THTTPRIO, THTTPSoapPascalInvoker, and THTTPSoapCppInvoker use TOPToSoapDomConvert to convert between remotable classes and their SOAP representation. TOPToSoapDomConvert has an Options property that specifies, among other things, whether to encode multiple references to the same object by value or as references. Individual remotable classes can supply an override to this default strategy when they are registered using the RegisterXSClass (see page 885) method.

ClassOptions returns how to handle multiple references of a specified class.

AClass is a class reference for the registered class that you want to know how to convert.

See Also

RegisterXSClass (see page 885)

Options (see page 2781)

IOPConvert (see page 2730)

THTTPSoapCppInvoker

THTTPSoapPascalInvoker

THTTPRIO

TOPToSoapDomConvert (see page 2747)

23.1.9.4.2 ClassToURI Method

This is the overview for the ClassToURI method overload.

Overload List

	Name	Description
	TRemotableTypeRegistry.ClassToURI (TClass, InvString, InvString) (see page 878)	Returns the namespace URI associated with a specified remotable class.
	TRemotableTypeRegistry.ClassToURI (TClass, InvString, InvString, Boolean, Boolean) (see page 878)	Returns the namespace URI associated with a specified remotable class.

23.1.9.4.2.1 TRemotableTypeRegistry.ClassToURI Method (TClass, InvString, InvString)

Returns the namespace URI associated with a specified remotable class.

Pascal

```
function ClassToURI(AClass: TClass; var URI: InvString; var Name: InvString; var IsScalar:
Boolean; tryToRegister: Boolean = True): Boolean; overload;
function ClassToURI(AClass: TClass; var URI: InvString; var Name: InvString): Boolean;
overload;
```

C++

```
__fastcall Boolean ClassToURI(TClass AClass, InvString URI, InvString Name, Boolean IsScalar,
Boolean tryToRegister = True);
__fastcall Boolean ClassToURI(TClass AClass, InvString URI, InvString Name);
```

Description

ClassToURI looks up a registered remotable class and retrieves its namespace URI and typename. The association of the class and its namespace URI and typename is established when you call RegisterXSClass (see page 885) to register the remotable class.

AClass is a class reference for the remotable class you want to look up.

URI returns the namespace URI that is associated with the class.

Name returns the registered name of the type the class represents.

IsScalar indicates whether the class corresponds to a scalar type in a WSDL document.

tryToRegister indicates whether the remotable type registry should auto-register the class that AClass specifies if it is not already registered. This parameter is only used if the AutoRegisterNativeTypes (see page 891) property is true.

See Also

RegisterXSClass (see page 885)

URIToClass (see page 889)

AutoRegisterNativeTypes (see page 891)

23.1.9.4.3 TRemotableTypeRegistry.GetExternalPropName Method

Returns the name of the XML element that corresponds to a property of a registered remotable class.

Pascal

```
function GetExternalPropName(Info: PTypeInfo; InternalName: string): InvString;
```

C++

```
__fastcall InvString GetExternalPropName(PTypeInfo Info, UnicodeString InternalName);
```

Description

Use GetExternalPropName to obtain the name of the XML element that was registered as the XML schema analog of a property on a registered remotable class. The association between the element and the property must be previously registered by the RegisterExternalPropName (see page 883) method.

Info points to the type information of the remotable class that has the property.

InternalName is the name of the property from which you want to obtain the corresponding XML element name.

See Also

GetInternalPropName (see page 879)

RegisterExternalPropName (see page 883)

23.1.9.4.4 TRemotableTypeRegistry.GetInternalPropName Method

Returns the name of the property on a remotable class that corresponds to a specified XML element.

Pascal

```
function GetInternalPropName(Info: PTypeInfo; ExternalName: InvString): string;
```

C++

```
__fastcall UnicodeString GetInternalPropName(PTypeInfo Info, InvString ExternalName);
```

Description

Use GetInternalPropName to obtain the name of the property on a registered remotable class that corresponds to a specified XML element. The association between the element and the property must be previously registered by the RegisterExternalPropName (see page 883) method.

Info points to the type information of the remotable class that has the property.

ExternalName is the name of an XML element that is part of a complex type defined in an XML schema. The remotable class identified by Info must represent the complex type that includes ExternalName.

See Also

GetExternalPropName (see page 879)

RegisterExternalPropName (see page 883)

23.1.9.4.5 TRemotableTypeRegistry.GetURICount Method

Returns the number of entries in the remotable type registry.

Pascal

```
function GetURICount: Integer;
```

C++

```
__fastcall int GetURICount();
```

Description

GetURICount returns the number of entries in the remotable type registry. This value can be used as an upper bound for the index parameter of the GetURIMap (see page 880) method.

Each entry in the remotable type registry represents a registered class or a registered type. The RegisterXSClass (see page 885) method adds entries for classes, and the The RegisterXSInfo (see page 886) method adds entries for types. Each entry associates the class or type with a unique namespace URI.

See Also

GetURIMap (see page 880)

RegisterXSClass (see page 885)

RegisterXSInfo (see page 886)

23.1.9.4.6 GetURIMap Method

This is the overview for the GetURIMap method overload.

Overload List

	Name	Description
	TRemotableTypeRegistry.GetURIMap (Integer) (see page 880)	Returns an entry from the remotable type registry.
	TRemotableTypeRegistry.GetURIMap (Integer, TRemRegEntry) (see page 880)	Returns an entry from the remotable type registry.

23.1.9.4.6.1 TRemotableTypeRegistry.GetURIMap Method (Integer)

Returns an entry from the remotable type registry.

Pascal

```
function GetURIMap(Index: Integer): TRemRegEntry; overload;
function GetURIMap(Index: Integer; out RegEntry: TRemRegEntry): boolean; overload;
```

C++

```
__fastcall TRemRegEntry GetURIMap(int Index);
__fastcall boolean GetURIMap(int Index, TRemRegEntry RegEntry);
```

Description

GetURIMap returns an entry from the remotable type registry that is identified by its index. It is useful for operations that iterate over all the registered entries. Use the GetURICount (see page 879) method to obtain an upper bound for the index.

Index is the index of the entry you want to see.

Using the first syntax, GetURIMap returns a TRemRegEntry (see page 944) value that describes a registered class or type.

Using the second syntax, the RegEntry parameter returns the TRemRegEntry (see page 944) at the specified index. Using this syntax, GetURIMap returns true if Index is a valid index, false if Index is out of range.

The TRemRegEntry (see page 944) value that this method returns is used to represent both remotable classes, which are added using the RegisterXSClass (see page 885) method, and remotable types, which are added using the RegisterXSInfo (see page 886) method.

See Also

GetURICount (see page 879)

RegisterXSClass (see page 885)

23.1.9.4.7 TRemotableTypeRegistry.GetVarTypeFromXSD Method

Returns the variant type that corresponds to a registered remotable type.

Pascal

```
function GetVarTypeFromXSD(const URI: InvString; const TypeName: InvString): TVarType;
```

C++

```
__fastcall TVarType GetVarTypeFromXSD(const InvString URI, const InvString TypeName);
```

Description

GetVarTypeFromXSD returns the built-in Variant type that corresponds most naturally to a remotable type that was registered by a call to the RegisterXSInfo (see page 886) method.

URI is the namespace URI associated with the type when it was registered.

TypeName is the type name associated with the type when it was registered.

See Also

VariantToInfo (see page 890)

XSDToTypeInfo (see page 891)

URIToInfo (see page 890)

RegisterXSInfo (see page 886)

23.1.9.4.8 TRemotableTypeRegistry.GetXSDInfoForClass Method

Returns type name and namespace URI that correspond to a remotable class.

Pascal

```
procedure GetXSDInfoForClass(Info: PTypeInfo; var URI: InvString; var TypeName: InvString);
```

C++

```
__fastcall GetXSDInfoForClass(PTypeInfo Info, InvString URI, InvString TypeName);
```

Description

GetXSDInfoForClass returns the type name under which a remotable class is registered, along with the namespace URI in which that type name is defined. The class must be previously registered by a call to the RegisterXSClass (see page 885) method.

Info is the type info pointer for the registered remotable class.

URI returns the namespace URI that was associated with the class when it was registered.

TypeName returns the type name that was associated with the class when it was registered.

See Also

RegisterXSClass (see page 885)

XSDToTypeInfo (see page 891)

23.1.9.4.9 TRemotableTypeRegistry.InfoToURI Method

Returns the namespace URI and type name for a registered type.

Pascal

```
function InfoToURI(Info: PTypeInfo; var URI: InvString; var Name: InvString; var IsScalar:
Boolean; tryToRegister: Boolean = True): Boolean;
```

C++

```
__fastcall Boolean InfoToURI(PTypeInfo Info, InvString URI, InvString Name, Boolean IsScalar,
Boolean tryToRegister = True);
```

Description

InfoToURI returns the namespace URI and type name that is associated with the specified type description. This association is established when you register a type by calling the RegisterXSInfo (see page 886) method.

Info is a pointer to the runtime type information (RTTI) for the registered type.

URI returns the namespace URI that is associated with the type.

Name returns the registered name of the type.

IsScalar indicates whether the type corresponds to a scalar type in a WSDL document.

tryToRegister indicates whether the remotable type registry should auto-register the type that Info specifies if it is not already registered. This parameter is only used if the AutoRegisterNativeTypes (see page 891) property is true.

See Also

RegisterXSInfo (see page 886)

AutoRegisterNativeTypes (see page 891)

23.1.9.4.10 TRemotableTypeRegistry.IsClassScalar Method

Indicates whether a registered class represents a scalar type.

Pascal

```
function IsClassScalar(AClass: TClass): Boolean;
```

C++

```
__fastcall Boolean IsClassScalar(TClass AClass);
```

Description

IsClassScalar returns true if the specified class represents a scalar type. Classes that represent scalar types descend from the TRemotableXS (see page 892) class, and can convert their value to or from a string.

AClass identifes the class to check. It must be previously registered by a call to RegisterXSClass (see page 885).

See Also

RegisterXSClass (see page 885)

TRemotableXS (see page 892)

23.1.9.4.11 TRemotableTypeRegistry.Lock Method

Prevents other threads from accessing the type registry.

Pascal

```
procedure Lock; virtual;
```

C++

```
virtual __fastcall Lock();
```

Description

Lock uses a critical section to prevent simultaneous use of the type registry by multiple threads in the same application. Once one thread has called Lock to obtain access to the type registry, no other thread can return from the Lock method until the first thread has called the UnLock (see page 888) method to release its lock.

Lock is called automatically by the methods of TRemotableTypeRegistry (see page 874).

See Also

UnLock (see page 888)

23.1.9.4.12 TRemotableTypeRegistry.RegisterExternalPropName Method

Registers the correspondence between a property name on a remotable class and the corresponding element name in an XML schema.

Pascal

```
procedure RegisterExternalPropName(Info: PTypeInfo; const InternalName: string; const
ExternalName: InvString);
```

C++

```
__fastcall RegisterExternalPropName(PTypeInfo Info, const UnicodeString InternalName, const
InvString ExternalName);
```

Description

Use RegisterExternalPropName to create an association between the name of an element in an XML schema complex type and the corresponding property of the remotable class that represents that type. This allows the remotable class to represent complex types that include elements with names that are not valid (for example, keywords).

Info points to the type information of the remotable class that corresponds to a complex type defined in an XML schema.

InternalName is the name used for the property of the remotable class that corresponds to the element.

ExternalName is the name of the element that is part of the complex type that the XML schema defines.

23.1.9.4.13 TRemotableTypeRegistry.RegisterHolderClsMember Method

Registers a remotable type whose type information must be extracted from a holder class.

Pascal

```
procedure RegisterHolderClsMember(ClsTypeInfo: PTypeInfo; const URI: InvString = ''; const
Name: InvString = ''; const ExtName: InvString = '');
```

C++

```
__fastcall RegisterHolderClsMember(PTypeInfo ClsTypeInfo, const InvString URI = '', const
InvString Name = '', const InvString ExtName = '');
```

Description

Use RegisterHolderClsMember to register a type that is used as the type of the only published property of a holder class. This method is used primarily for C++ applications, which can contain types that require a holder class so that the compiler can extract their runtime type information (RTTI). If the holder class contains more than one published property, use the RegisterXSInfo (see page 886) method after extracting the type information from the property using the global GetClsMemberTypeInfo (see page 792) function.

Info points to the type information of the holder class that has a single published member whose type you want to register.

URI is the namespace URI to associate with the type that is registered. This should uniquely identify the type. If you omit this parameter or supply an empty string, RegisterXSInfo (see page 886) generates a namespace URI for you, based on the name of the type, the name of the unit in which it is declared, and the value of the global AppNameSpacePrefix (see page 955) variable. Note that the namespace URI must be the same in both client and server applications. In Delphi, if you do not explicitly supply a namespace URI, the class must be declared in a unit with the same name, and the value of AppNameSpacePrefix (see page 955) must be the same in both applications. In C++, the client application must always explicitly supply a namespace.

Name is the name of the type as it appears in native code.

ExtName is the name of the type as it appears in encoded method calls and WSDL documents.

See Also

GetClsMemberTypeInfo (see page 792)

RegisterXSInfo (see page 886)

23.1.9.4.14 RegisterSerializeOptions Method

This is the overview for the RegisterSerializeOptions method overload.

Overload List

	Name	Description
	TRemotableTypeRegistry.RegisterSerializeOptions (PTypeInfo, TSerializationOptions) (see page 884)	Registers a set of serialization options that should be used with a registered remotable class or type.
	TRemotableTypeRegistry.RegisterSerializeOptions (TClass, TSerializationOptions) (see page 884)	Registers a set of serialization options that should be used with a registered remotable class or type.

23.1.9.4.14.1 TRemotableTypeRegistry.RegisterSerializeOptions Method (PTypeInfo, TSerializationOptions)

Registers a set of serialization options that should be used with a registered remotable class or type.

Pascal

```
procedure RegisterSerializeOptions(Info: PTypeInfo; SerialOpt: TSerializationOptions);
overload;
procedure RegisterSerializeOptions(AClass: TClass; SerialOpt: TSerializationOptions); overload;
```

C++

```
__fastcall RegisterSerializeOptions(PTypeInfo Info, TSerializationOptions SerialOpt);
__fastcall RegisterSerializeOptions(TClass AClass, TSerializationOptions SerialOpt);
```

Description

RegisterSerializeOptions associates a registered remotable class or type with a set of flags that indicate how to encode or decode its value. Typically, this method is called by the WSDL importer.

AClass identifes the class for which you are supplying serialization options. It must be previously registered by a call to RegisterXSClass (see page 885).

Info is the runtime type information for a type when you are supplying serialization options for a type rather than a class. It must be previously registered by a call to RegisterXSInfo (see page 886).

SerialOpt is the set of flags that customize how the class is encoded or decoded.

See Also

SerializationOptions (see page 873)

SerializeOptions (see page 887)

RegisterXSClass (see page 885)

RegisterXSInfo (see page 886)

Defining and Using SOAP Headers

23.1.9.4.15 TRemotableTypeRegistry.RegisterXSClass Method

Registers a remotable class.

Pascal

```
procedure RegisterXSClass(AClass: TClass; const URI: InvString = ''; const Name: InvString =
'''; const ExtName: InvString = ''; IsScalar: Boolean = False; MultiRefOpt: TObjMultiOptions =
ocDefault);
```

C++

```
__fastcall RegisterXSClass(TClass AClass, const InvString URI = '', const InvString Name = '',
const InvString ExtName = '', Boolean IsScalar = False, TObjMultiOptions MultiRefOpt =
ocDefault);
```

Description

Call RegisterXSClass to register a remotable class so that it can be communicated between client applications and a Web Service provider. All remotable classes are compiled with runtime type information (RTTI) and have a virtual constructor that the remotable type registry recognizes and uses to supply class instances.

When the remotable class is a TRemotable (see page 868) descendant, it can be used to represent non-scalar types for parameters and return values on an invokable interface. When it is an ERemotableException (see page 810) descendant, it allows the Web Service application to send clients a fault packet that includes all the published properties of the exception class.

AClass is the class reference for the remotable class. In Delphi, you specify this in code using the class name. In C++, you can obtain this class reference for a class using the __classid keyword.

URI is the namespace URI to associate with that class. This should uniquely identify the class. If you omit this parameter (or supply an empty string), RegisterXSClass generates a namespace URI for you, based on the name of the class, the name of the unit in which it is declared, and the value of the global AppNameSpacePrefix (see page 955) variable. Note that the namespace URI must be the same in both client and server applications. Thus, if you do not explicitly supply a namespace URI, the class must be declared in a unit with the same name, and the value of AppNameSpacePrefix (see page 955) must be the same in both applications.

Name is the name of a type that the class represents. This is used for the mapping between the class and its definition in a WSDL document.

ExtName is the same as Name.

IsScalar indicates whether the class represents a scalar type. Remotable classes that represent scalar types must implement the XSToNative and NativeToXS methods.

MultiRefOpt indicates how multiple references to the same instance of the class should be handled when converting them to their SOAP representation. For more information, see the ClassOptions (see page 877) method.

See Also

TRemotable (see page 868)

ERemotableException (see page 810)

AppNameSpacePrefix (see page 955)

UnRegisterXSClass (see page 888)

ClassOptions (see page 877)

Creating Custom Exception Classes for Web Services

Registering Nonscalar Types

23.1.9.4.16 TRemotableTypeRegistry.RegisterXSInfo Method

Registers a remotable type.

Pascal

```
procedure RegisterXSInfo(Info: PTypeInfo; const URI: InvString = ''; const Name: InvString =
'''; const ExtName: InvString = '');
```

C++

```
__fastcall RegisterXSInfo(PTypeInfo Info, const InvString URI = '', const InvString Name = '',
const InvString ExtName = '');
```

Description

Call RegisterXSInfo to register a type that requires special marshaling but does not need to be represented by a TRemotable (see page 868) descendant before it can be marshaled for transport to or from a Web Services provider. Such types include dynamic arrays such as the dynamic array types defined in the Types unit (Delphi) or sysdyn.h (C++), enumerated types, and boolean types (Delphi) or types defined using a typedef statement (C++).

Info is the TypeInfo pointer for the type. It can be obtained using the global TypeInfo function (Delphi) or the __delphirtti keyword (C++). In C++, it is sometimes obtained from a global function such as GetClsMemberTypeInfo (see page 792), which extracts the type information from a holder class.

URI is the namespace URI to associate with that type. This should uniquely identify the type. If you omit this parameter or supply an empty string, RegisterXSInfo generates a namespace URI for you, based on the name of the type, the name of the unit in which it is declared, and the value of the global AppNameSpacePrefix (see page 955) variable. Note that the namespace URI must be the same in both client and server applications. In Delphi, if you do not explicitly supply a namespace URI, the class must be declared in a unit with the same name, and the value of AppNameSpacePrefix (see page 955) must be the same in both applications. In C++, the client application must always explicitly supply a namespace.

Name is the name of the type as it appears in native code.

ExtName is the name of the type as it appears in encoded method calls and WSDL documents.

See Also

AppNameSpacePrefix (see page 955)

UnRegisterXSInfo (see page 889)

GetClsMemberTypeInfo (see page 792)

Registering Nonscalar Types

23.1.9.4.17 SerializeOptions Method

This is the overview for the SerializeOptions method overload.

Overload List

	Name	Description
	TRemotableTypeRegistry.SerializeOptions (PTypeInfo) (see page 887)	Returns the serialization options for a registered remotable class or type.
	TRemotableTypeRegistry.SerializeOptions (TClass) (see page 887)	Returns the serialization options for a registered remotable class or type.

23.1.9.4.17.1 TRemotableTypeRegistry.SerializeOptions Method (PTypeInfo)

Returns the serialization options for a registered remotable class or type.

Pascal

```
function SerializeOptions(Info: PTypeInfo): TSerializationOptions; overload;
function SerializeOptions(AClass: TClass): TSerializationOptions; overload;
```

C++

```
__fastcall TSerializationOptions SerializeOptions(PTypeInfo Info);
__fastcall TSerializationOptions SerializeOptions(TClass AClass);
```

Description

SerializeOptions is used internally to retrieve the flags that describe how a remotable class or type is encoded and decoded. These flags are associated with the remotable class or type by calling the RegisterSerializeOptions (see page 884) method.

AClass is the registered class for which you want to retrieve serialization options.

Info is the runtime type information for the registered type for which you want to retrieve serialization options.

SerializeOptions returns the value of SerializationOptions (see page 940) for the class that AClass specifies or the registered options for the type that Info specifies..

See Also

SerializationOptions (see page 873)

RegisterSerializeOptions (see page 884)

23.1.9.4.18 TRemotableTypeRegistry.TypeInfoToXSD Method

Returns the type name and namespace URI registered with a remotable class or type.

Pascal

```
function TypeInfoToXSD(Info: PTypeInfo; var URI: InvString; var TypeName: InvString): Boolean;
```

C++

```
__fastcall Boolean TypeInfoToXSD(PTypeInfo Info, InvString URI, InvString TypeName);
```

Description

TypInfoToXSD returns a type name and namespace URI from a type information pointer. The type information can describe either a remotable class that was previously registered by a call to RegisterXSClass (see page 885), or a remotable type that was previously registered by a call to RegisterXSInfo (see page 886).

Info is the type information pointer that describes the class or type.

URI returns the namespace URI that was associated with the class or type when it was registered.

TypeName returns the type name that was associated with the class or type when it was registered.

See Also

XSDToTypeInfo (see page 891)

RegisterXSInfo (see page 886)

RegisterXSClass (see page 885)

InfoToURI (see page 882)

23.1.9.4.19 TRemotableTypeRegistry.UnLock Method

Allows other threads to access the type registry after it was locked.

Pascal

```
procedure UnLock; virtual;
```

C++

```
virtual __fastcall UnLock();
```

Description

UnLock exits the critical section that the Lock (see page 883) method entered. It reenables other threads access to the type registry after a call to the Lock (see page 883) method.

Every call to the Lock (see page 883) method must be paired by a call to UnLock when the locking thread is finished with the type registry. The methods of TRemotableTypeRegistry (see page 874) that call the Lock (see page 883) method also call the UnLock method when they are finished.

See Also

Lock (see page 883)

23.1.9.4.20 TRemotableTypeRegistry.UnRegisterXSClass Method

Unregisters a remotable class.

Pascal

```
procedure UnRegisterXSClass(AClass: TClass);
```

C++

```
__fastcall UnRegisterXSClass(TClass AClass);
```

Description

Call UnRegisterXSClass to Unregister a remotable class that was previously registered by a call to the RegisterXSClass (see page 885) method. In Delphi, this call is made in the finalization section of the unit that implements the remotable class. In C++, it is made from a method that uses the pragma exit directive in the unit that implements the remotable class.

AClass is the class reference for the remotable class. In Delphi code, you specify this using the class name. In C++, you can obtain this class reference for a class using the __classid keyword.

See Also

TRemotable (see page 868)

ERemotableException (see page 810)

RegisterXSClass (see page 885)

23.1.9.4.21 TRemotableTypeRegistry.UnRegisterXSInfo Method

Unregisters a remotable type.

Pascal

```
procedure UnRegisterXSInfo(Info: PTypeInfo);
```

C++

```
__fastcall UnRegisterXSInfo(PTypeInfo Info);
```

Description

Call UnRegisterXSInfo to unregister a type that was previously registered (directly or indirectly) by a call to the RegisterXSInfo (see page 886) method.

Info is the TypeInfo pointer for the type. It is the same as the Info parameter of the RegisterXSInfo (see page 886) method.

See Also

RegisterXSInfo (see page 886)

23.1.9.4.22 URIToClass Method

This is the overview for the URIToClass method overload.

Overload List

	Name	Description
	TRemotableTypeRegistry.URIToClass (InvString, InvString) (see page 889)	Returns the class associated with a specified namespace URI.
	TRemotableTypeRegistry.URIToClass (InvString, InvString, Boolean) (see page 889)	Returns the class associated with a specified namespace URI.

23.1.9.4.22.1 TRemotableTypeRegistry.URIToClass Method (InvString, InvString)

Returns the class associated with a specified namespace URI.

Pascal

```
function URIToClass(const URI: InvString; const Name: InvString; var IsScalar: Boolean):
TClass; overload;
function URIToClass(const URI: InvString; const Name: InvString): TClass; overload;
```

C++

```
__fastcall TClass URIToClass(const InvString URI, const InvString Name, Boolean IsScalar);
__fastcall TClass URIToClass(const InvString URI, const InvString Name);
```

Description

URIToClass looks up a registered remotable class given its namespace URI and typename.

URI is the namespace URI associated with the class. This association is established by the RegisterXSClass (see page 885) method.

Name is the name of the type that the class represents. If the class does not have a corresponding type name, Name should be an empty string.

IsScalar returns true if the class represents a scalar type, false otherwise.

URIToClass returns the class reference for the class identified by URI (and Name).

See Also

RegisterXSClass (see page 885)

ClassToURI (see page 878)

23.1.9.4.23 TRemotableTypeRegistry.URIToInfo Method

Returns the type information pointer for a registered remotable class or type.

Pascal

```
function URIToInfo(const URI: InvString; const Name: InvString): PTypeInfo;
```

C++

```
__fastcall PTypeInfo URIToInfo(const InvString URI, const InvString Name);
```

Description

URIToInfo looks up a registered remotable class or type given its namespace URI and typename. It returns the type information pointer that describes the type or class.

URI is the namespace URI associated with the class or type. This association is established by the RegisterXSClass (see page 885) or RegisterXSInfo (see page 886) method.

Name is the type name that was associated with the class or type when it was registered.

Tip: To look up the type information for a namespace and type that may possibly represent a built-in XML schema type, use the XSDToTypeInfo (see page 891) method instead.

See Also

RegisterXSClass (see page 885)

InfoToURI (see page 882)

RegisterXSInfo (see page 886)

23.1.9.4.24 TRemotableTypeRegistry.VariantToInfo Method

Returns the TypeInfo pointer for the type of a specified Variant's data.

Pascal

```
function VariantToInfo(const V: Variant; TryAllSchema: Boolean): PTypeInfo;
```

C++

```
__fastcall PTypeInfo VariantToInfo(const Variant V, Boolean TryAllSchema);
```

Description

VariantToInfo returns the TypeInfo pointer of the scalar type that represents the value of a Variant. If the Variant's value is not a scalar type that can be marshaled without using a registered object or type, VariantToInfo raises an ETypeRegistryException (see page 813) exception.

V is the Variant for which you want the corresponding TypeInfo pointer.

TryAllSchema specifies whether VariantToInfo should check for corresponding scalar type definitions in the XML schema associated with this application, or whether it should check all the possible sets of XML schema types that are defined in the

xmlschema unit.

See Also

VarType

ETypeRegistryException (see page 813)

23.1.9.4.25 TRemotableTypeRegistry.XSDToTypeInfo Method

Returns the type information pointer associated with a specified type name.

Pascal

```
function XSDToTypeInfo(const URI: InvString; const TypeName: InvString): PTypeInfo;
```

C++

```
__fastcall PTypeInfo XSDToTypeInfo(const InvString URI, const InvString TypeName);
```

Description

XSDToTypeInfo looks up a namespace URI and typename, and returns the type information pointer that describes the corresponding Object Pascal type or class. Unlike the URIToInfo (see page 890) method, XSDToTypeInfo can look up built-in type correspondences that do not need to be registered.

URI is the namespace URI for the type.

TypeName is the type name to look up.

See Also

RegisterXSClass (see page 885)

URIToInfo (see page 890)

RegisterXSInfo (see page 886)

23.1.9.5 TRemotableTypeRegistry Properties

The properties of the TRemotableTypeRegistry class are listed here.

Public Properties

	Name	Description
	AutoRegisterNativeTypes (see page 891)	Specifies whether types that are used in an invokable interface or by a remotable class are automatically registered.

23.1.9.5.1 TRemotableTypeRegistry.AutoRegisterNativeTypes Property

Specifies whether types that are used in an invokable interface or by a remotable class are automatically registered.

Pascal

```
property AutoRegisterNativeTypes: Boolean;
```

C++

```
__property Boolean AutoRegisterNativeTypes;
```

Description

When AutoRegisterNativeTypes is true (the default), type definitions are automatically registered when the invokable interface or remotable class that uses them is registered. That is, when you register an invokable interface with the invocation registry or

when you register a remotable class with the remotable type registry, the registry checks whether it makes use of types that are not yet registered, and if so, registers them as well. Note that you can still provide your own registration (for example, to control the external name or namespace) by registering the type before the call that registers the interface or remotable class.

See Also

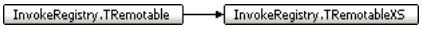
RegisterXSInfo (see page 886)

RegisterXSClass (see page 885)

23.1.10 TRemotableXS Class

TRemotableXS is the base class for classes that represent a non-native scalar type on an invokable interface in a Web Service application.

Class Hierarchy



Pascal

```
TRemotableXS = class(TRemotable);
```

C++

```
class TRemotableXS : public TRemotable;
```

File

InvokeRegistry

Description

Use TRemotableXS as a base class when defining a class to represent scalar values from a WSDL document that have no corresponding Object Pascal type. When creating a TRemotableXS descendant, override the XSToNative (see page 895) and NativeToXS (see page 893) methods to indicate how the value of the class can be converted to and from the string representation that is used to encode the value for transport.

Both the client application and server application must register a TRemotableXS descendant with the remotable class registry before they can use it. To register a remotable class, use the RegisterXSClass method of the object that the global RemTypeRegistry (see page 931) function returns.

On server applications, instances of TRemotableXS descendants that are input parameters on an invokable interface are automatically created when the method call is unmarshaled and automatically freed after any output parameters or return value are marshaled so that they can be sent to the client. When a TRemotableXS descendant is created in a method that was called remotely using an invokable interface, the instance is automatically freed after the value of the TRemotable (see page 868) descendant is marshaled for transport back to the client application.

The caller of an invokable interface (client) is responsible for creating any TRemotableXS instances that it passes as input parameters, and for freeing any TRemotableXS instances it creates or that are returned as output parameters or method results.

See Also

RemTypeRegistry (see page 931)

TRemotableTypeRegistry (see page 874)

IInvokable

Using Remotable Objects

23.1.10.1 TRemotableXS Members

The following tables list the members exposed by TRemotableXS.

Protected Fields

	Name	Description
	FSerializationOptions (see page 871)	This is FSerializationOptions, a member of class TRemotable.

Public Methods

	Name	Description
	Create (see page 870)	Creates and initializes a TRemotable (see page 868) object.
	Destroy (see page 870)	Frees the memory associated with the TRemotable (see page 868) object.
	NativeToXS (see page 893)	Returns a string that represents the value of a TRemotableXS (see page 892) instance.
	ObjectToSOAP (see page 894)	Converts this remotable object to its SOAP representation.
	SOAPToObject (see page 894)	Initializes this TRemotableXS (see page 892) instance to represent the value of its SOAP representation.
	XSToNative (see page 895)	Assigns the value of a string to the TRemotableXS (see page 892) instance.

Public Properties

	Name	Description
	DataContext (see page 873)	Specifies the object that controls the lifetime of a TRemotable (see page 868) instance.
	SerializationOptions (see page 873)	Indicates how the value of the TRemotable (see page 868) descendant encodes its value.

23.1.10.2 TRemotableXS Methods

The methods of the TRemotableXS class are listed here.

Public Methods

	Name	Description
	NativeToXS (see page 893)	Returns a string that represents the value of a TRemotableXS (see page 892) instance.
	ObjectToSOAP (see page 894)	Converts this remotable object to its SOAP representation.
	SOAPToObject (see page 894)	Initializes this TRemotableXS (see page 892) instance to represent the value of its SOAP representation.
	XSToNative (see page 895)	Assigns the value of a string to the TRemotableXS (see page 892) instance.

23.1.10.2.1 TRemotableXS.NativeToXS Method

Returns a string that represents the value of a TRemotableXS (see page 892) instance.

Pascal

```
function NativeToXS: InvString; virtual; abstract;
```

C++

```
virtual __fastcall InvString NativeToXS() = 0;
```

Description

Override NativeToXS when implementing a TRemotableXS (see page 892) descendant that represents a non-native scalar type. NativeToXS returns a string that represents the value of the TRemotableXS (see page 892) instance. This string is used when marshaling the value of the TRemotableXS (see page 892) descendant.

Classes that implement NativeToXS must also override the XSToNative (see page 895) method so that it performs the converse transformation.

See Also

XSToNative (see page 895)

Using Remotable Objects

23.1.10.2.2 TRemotableXS.ObjectToSOAP Method

Converts this remotable object to its SOAP representation.

Pascal

```
function ObjectToSOAP(RootNode: IXMLNode; ParentNode: IXMLNode; const ObjConverter:
IObjConverter; const Name: InvString; const URI: InvString; ObjConvOpts:
TObjectConvertOptions; out RefID: InvString): IXMLNode; override;
```

C++

```
virtual __fastcall IXMLNode ObjectToSOAP(IXMLNode RootNode, IXMLNode ParentNode, const
IObjConverter ObjConverter, const InvString Name, const InvString URI, TObjectConvertOptions
ObjConvOpts, InvString RefID);
```

Description

Override ObjectToSOAP to customize the way your application converts a TRemotableXS (see page 892) descendant to its SOAP representation. In TRemotableXS (see page 892), ObjectToSOAP uses the default conversion, replacing the text of the node with the value that the NativeToXS (see page 893) method supplies.

RootNode is an ancestor of the generated node to which information about the generated node can be added as attributes.

ParentNode is the node in the SOAP representation that should act as a parent to the node that ObjectToSOAP generates.

ObjConverter is the interface of the default converter that translates between remotable objects and their SOAP representation. This is the interface to a TOPToSoapDomConvert instance.

Name is the name of the node that is generated.

URI is the URI for the namespace in which Name is defined.

ObjConvOpts are flags that customize the way the node is serialized.

RefID returns a reference ID for the node. This is used if the node is multi-referenced.

ObjectToSOAP returns the XML for the newly generated node.

See Also

SOAPToObject (see page 894)

NativeToXS (see page 893)

23.1.10.2.3 TRemotableXS.SOAPToObject Method

Initializes this TRemotableXS (see page 892) instance to represent the value of its SOAP representation.

Pascal

```
procedure SOAPToObject(const RootNode: IXMLNode; const Node: IXMLNode; const ObjConverter:
IObjConverter); override;
```

C++

```
virtual __fastcall SOAPToObject(const IXMLNode RootNode, const IXMLNode Node, const
```

```
IObjConverter ObjConverter);
```

Description

SOAPToObject initializes the published properties of this object instance so that they reflect the header's SOAP representation. Override this method to customize the way your application creates a scalar value from its XML representation.

RootNode is an ancestor node of the node for this object that can contain information about the remotable type's value.

Node is the node in the SOAP envelope that represents the remotable type's value.

ObjConverter is the interface of the default converter that translates between remotable objects and their SOAP representation. This is the interface to a TOPToSoapDomConvert instance.

In TRemotableXS (see page 892), SOAPToObject calls the XSToNative (see page 895) method, passing in the text of the specified node.

See Also

ObjectToSOAP (see page 894)

XSToNative (see page 895)

23.1.10.2.4 TRemotableXS.XSToNative Method

Assigns the value of a string to the TRemotableXS (see page 892) instance.

Pascal

```
procedure XSToNative(Data: InvString); virtual; abstract;
```

C++

```
virtual __fastcall XSToNative(InvString Data) = 0;
```

Description

Override XSToNative when implementing a TRemotableXS (see page 892) descendant that represents a non-native scalar type. NativeToXS (see page 893) interprets a string that represents the value of the TRemotableXS (see page 892) instance and uses it to make any necessary changes so that the TRemotableXS (see page 892) object reflects that value. It reverses the transformation that the NativeToXS (see page 893) method performs.

Data is a string that represents the value of the TRemotableXS (see page 892) instance.

See Also

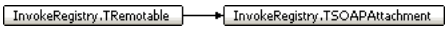
NativeToXS (see page 893)

Using Remotable Objects

23.1.11 TSOAPAttachment Class

TSOAPAttachment represents an attachment that is included with SOAP messages that represent method calls in a Web Service application.

Class Hierarchy



Pascal

```
TSOAPAttachment = class(TRemotable);
```

C++

```
class TSOAPAttachment : public TRemotable;
```

File

InvokeRegistry

Description

Use TSOAPAttachment to receive or send an attachment as a parameter or return value of a method on an invokable interface. When a SOAP method call requires an attachment, it is marshaled as a MIME multipart form, where any attachments are included as additional parts.

Warning: Some Web Services (such as .Net) use DIME multipart forms rather than MIME multipart forms for attachments. TSOAPAttachment will not work with such services.

Note: Only use TSOAPAttachment and its descendants as the type for parameters or the return value of a method of an invokable interface. TSOAPAttachment can't be used as part of a more complex type, such as the type of a published property on a remotable object.

When sending an attachment, use the SourceString (see page 908) property or the SetSourceFile (see page 902) or SetSourceStream (see page 903) method to supply the attachment data. Use the ContentType (see page 905) property to indicate the type (encoding) of the attachment data.

When receiving an attachment, the attachment is automatically stored in a temporary file, whose name is given by the CacheFile (see page 904) property. You can copy the contents of this temporary file using the SaveToFile (see page 901) or SaveToStream (see page 901) method. To determine the format of the data, read the ContentType (see page 905) property. You can read any of the other headers for the part of the multipart form that contained the attachment using the Headers (see page 906) property. By default, when the TSOAPAttachment object is freed, it deletes the temporary file that contains the attachment data that was received. You can use the CacheFilePersist (see page 905) property to prevent this file from being deleted.

Note: If you use TSOAPAttachment as a base class for a customized attachment class, it is marshaled as a TSOAPAttachment value. That is, the application that receives the attachment sees it only as a TSOAPAttachment instance, not as an instance of the descendant class.

The application must register TSOAPAttachment with the remotable class registry before it can use this class. To register TSOAPAttachment or one of its descendants, use the RegisterXSClass method of the object that the global RemTypeRegistry (see page 931) function returns. When you import a WSDL document, the call to register TSOAPAttachment is automatically added if any of the interfaces use attachments.

The caller of an invokable interface (client) is responsible for creating any TSOAPAttachment instances that it passes as input parameters, and for freeing any TSOAPAttachment instances it creates or that are returned as output parameters or method results.

In Web Service applications (servers), any TSOAPAttachment instances that represent parameters or return values are added to the data context, which frees them as necessary.

See Also

RemTypeRegistry (see page 931)

TRemotableTypeRegistry (see page 874)

IInvokable

Using Remotable Objects

23.1.11.1 TSOAPAttachment Members

The following tables list the members exposed by TSOAPAttachment.

Protected Fields

	Name	Description
	FSerializationOptions (see page 871)	This is FSerializationOptions, a member of class TRemotable.

Protected Methods

	Name	Description
V	GetSourceStream (see page 899)	This is GetSourceStream, a member of class TSOAPAttachment.
V	InternalSetCacheFile (see page 900)	This is InternalSetCacheFile, a member of class TSOAPAttachment.
	InternalSetSourceStream (see page 900)	This is InternalSetSourceStream, a member of class TSOAPAttachment.
V	SetOwnership (see page 902)	This is SetOwnership, a member of class TSOAPAttachment.
V	SetSourceString (see page 903)	This is SetSourceString, a member of class TSOAPAttachment.

Public Methods

	Name	Description
V	Create (see page 897)	Creates and initializes a TSOAPAttachment (see page 895) object.
V	Destroy (see page 898)	Frees the memory associated with the TSOAPAttachment (see page 895) object.
	Init (see page 899)	Initializes the properties of a TSOAPAttachment (see page 895) instance.
V	ObjectToSOAP (see page 900)	Converts this attachment object to its SOAP representation.
	SaveToFile (see page 901)	Saves the contents of the attachment to a specified file.
	SaveToStream (see page 901)	Saves the contents of the attachment to a specified file.
V	SetSourceFile (see page 902)	Specifies a file from which TSOAPAttachment (see page 895) can read the content of the attachment.
V	SetSourceStream (see page 903)	Specifies a stream from which TSOAPAttachment (see page 895) can read the content of the attachment.
V	SOAPToObject (see page 903)	Initializes this TSOAPAttachment (see page 895) instance to represent the value of its SOAP representation.

Public Properties

	Name	Description
R	CacheFile (see page 904)	Identifies the temporary file where the attachment is saved.
	CacheFilePersist (see page 905)	Indicates whether the temporary file that contains the attachment data is deleted when TSOAPAttachment (see page 895) is freed.
	ContentType (see page 905)	Specifies the MIME type of the attachment data.
	DataContext (see page 873)	Specifies the object that controls the lifetime of a TRemotable (see page 868) instance.
	Encoding (see page 906)	Indicates the encoding (character set) of the body of the attachment.
R	Headers (see page 906)	Indicates the values of the headers received with the attachment.
	Ownership (see page 907)	Indicates whether the TSOAPAttachment (see page 895) object should free its source stream.
R	SerializationOptions (see page 873)	Indicates how the value of the TRemotable (see page 868) descendant encodes its value.
R	SourceStream (see page 907)	Indicates a stream from which to read the content of an attachment that the application sends.
	SourceString (see page 908)	Specifies a string that is sent as the value of the attachment.

23.1.11.2 TSOAPAttachment.Create Constructor

Creates and initializes a TSOAPAttachment (see page 895) object.

Pascal

```
constructor Create; override;
```

C++

```
virtual __fastcall TSOAPAttachment();
```

Description

Call Create to instantiate a TSOAPAttachment (see page 895) object for an attachment you wish to send. When an application receives a SOAP attachment, an instance of TSOAPAttachment (see page 895) is created automatically to represent that attachment.

23.1.11.3 TSOAPAttachment.Destroy Destructor

Frees the memory associated with the TSOAPAttachment (see page 895) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TSOAPAttachment();
```

Description

Do not call Destroy to free a TSOAPAttachment (see page 895) object. Instead, use the Free method, which checks for a nil reference before calling Destroy.

On server applications, there is typically no need to explicitly free a TSOAPAttachment (see page 895) instance (for example, one created as a return value or output parameter). By default, when a TSOAPAttachment (see page 895) descendant is created in a method that was called remotely using an invokable interface, it is added to a data context (the value of the DataContext (see page 873) property). As long as the remotable object belongs to the data context, the data context handles freeing the object. Similarly, TSOAPAttachment (see page 895) instances that are passed in as a parameter belong to the data context.

On client applications, TSOAPAttachment (see page 895) instances created and passed to an invokable interface or returned as a parameter or result of an invokable interface must be freed by the calling application.

See Also

DataContext (see page 873)

System::TObject::Free

23.1.11.4 TSOAPAttachment Methods

The methods of the TSOAPAttachment class are listed here.

Protected Methods

	Name	Description
	GetSourceStream (see page 899)	This is GetSourceStream, a member of class TSOAPAttachment.
	InternalSetCacheFile (see page 900)	This is InternalSetCacheFile, a member of class TSOAPAttachment.
	InternalSetSourceStream (see page 900)	This is InternalSetSourceStream, a member of class TSOAPAttachment.
	SetOwnership (see page 902)	This is SetOwnership, a member of class TSOAPAttachment.
	SetSourceString (see page 903)	This is SetSourceString, a member of class TSOAPAttachment.

Public Methods

	Name	Description
	Init (see page 899)	Initializes the properties of a TSOAPAttachment (see page 895) instance.
V	ObjectToSOAP (see page 900)	Converts this attachment object to its SOAP representation.
	SaveToFile (see page 901)	Saves the contents of the attachment to a specified file.
	SaveToStream (see page 901)	Saves the contents of the attachment to a specified file.
V	SetSourceFile (see page 902)	Specifies a file from which TSOAPAttachment (see page 895) can read the content of the attachment.
V	SetSourceStream (see page 903)	Specifies a stream from which TSOAPAttachment (see page 895) can read the content of the attachment.
V	SOAPToObject (see page 903)	Initializes this TSOAPAttachment (see page 895) instance to represent the value of its SOAP representation.

23.1.11.4.1 TSOAPAttachment.GetSourceStream Method

Pascal

```
function GetSourceStream: TStream; virtual;
```

C++

```
virtual __fastcall TStream GetSourceStream();
```

Description

This is GetSourceStream, a member of class TSOAPAttachment.

23.1.11.4.2 TSOAPAttachment.Init Method

Initializes the properties of a TSOAPAttachment (see page 895) instance.

Pascal

```
procedure Init(ATempLocation: string; AHeaders: TStrings; AContentType: String; AEncoding:
InvString);
```

C++

```
__fastcall Init(UnicodeString ATempLocation, TStrings AHeaders, UnicodeString AContentType,
InvString AEncoding);
```

Description

Init sets the values of the CacheFile (see page 904), Headers (see page 906), ContentType (see page 905), and Encoding (see page 906) properties to the values supplied as parameters. It is used internally to initialize a TSOAPAttachment (see page 895) instance to describe an attachment that the application received.

ATempLocation is the name of the temporary file where the attachment has been stored. It becomes the value of the CacheFile (see page 904) property.

AHeaders lists the headers from the portion of the multipart form that represented this attachment. It becomes the value of the Headers (see page 906) property.

AContentType is the content-type of the attachment. It becomes the value of the ContentType (see page 905) property.

AEncoding is the character set encoding used for the attachment. It becomes the value of the Encoding (see page 906) property.

See Also

CacheFile (see page 904)

Headers (see page 906)

ContentType (see page 905)

Encoding (see page 906)

23.1.11.4.3 TSOAPAttachment.InternalSetCacheFile Method

Pascal

```
procedure InternalSetCacheFile(const Value: string); virtual;
```

C++

```
virtual __fastcall InternalSetCacheFile(const UnicodeString Value);
```

Description

This is InternalSetCacheFile, a member of class TSOAPAttachment.

23.1.11.4.4 TSOAPAttachment.InternalSetSourceStream Method

Pascal

```
procedure InternalSetSourceStream(const Value: TStream; const Ownership: TStreamOwnership =
soReference);
```

C++

```
__fastcall InternalSetSourceStream(const TStream Value, const TStreamOwnership Ownership =
soReference);
```

Description

This is InternalSetSourceStream, a member of class TSOAPAttachment.

23.1.11.4.5 TSOAPAttachment.ObjectToSOAP Method

Converts this attachment object to its SOAP representation.

Pascal

```
function ObjectToSOAP(RootNode: IXMLNode; ParentNode: IXMLNode; const ObjConverter:
IObjConverter; const Name: InvString; const URI: InvString; ObjConvOpts:
TObjectConvertOptions; out RefID: InvString): IXMLNode; override;
```

C++

```
virtual __fastcall IXMLNode ObjectToSOAP(IXMLNode RootNode, IXMLNode ParentNode, const
IObjConverter ObjConverter, const InvString Name, const InvString URI, TObjectConvertOptions
ObjConvOpts, InvString RefID);
```

Description

ObjectToSOAP customizes the way an application converts a TSOAPAttachment (see page 895) object to its SOAP representation. In TSOAPAttachment (see page 895), ObjectToSOAP calls the inherited method, adding ocoDontSerializeProps and ocoDontPutTypeAttr to the ObjConvOpts flags so that the converter ignores the published properties and does not add type information to the generated node. It then generates a unique identifier for the attachment and calls the converter's AddAttachment method to properly generate the SOAP representation of the attachment.

RootNode is an ancestor of the generated node to which information about the node can be added as attributes.

ParentNode is the node in the SOAP representation that should act as a parent to the node that ObjectToSOAP generates.

ObjConverter is the interface of the default converter that translates between remotable objects and their SOAP representation. This is the interface to a TOPToSoapDomConvert instance.

Name is the name of the node that is generated for the attachment.

URI is the URI for the namespace in which Name is defined.

ObjConvOpts are flags that customize the way the node is serialized.

RefID returns a reference ID for the node. This is used if the node is multi-referenced.

ObjectToSOAP returns the XML for the newly generated node.

See Also

ObjInstanceToSOAP (see page 922)

AddAttachment (see page 921)

SOAPToObject (see page 903)

23.1.11.4.6 TSOAPAttachment.SaveToFile Method

Saves the contents of the attachment to a specified file.

Pascal

```
procedure SaveToFile(AFileName: string);
```

C++

```
__fastcall SaveToFile(UnicodeString AFileName);
```

Description

Call SaveToFile to save the contents of the attachment to a specified file. This content can come from

The temporary file that the CacheFile (see page 904) property indicates.

The stream that the SourceStream (see page 907) property specifies.

The file that the SetSourceFile (see page 902) method supplies.

The string that the SourceString (see page 908) property specifies.

AFileName is the name of the file to which SaveToFile writes the content of the attachment.

See Also

CacheFile (see page 904)

SourceStream (see page 907)

SourceString (see page 908)

SaveToStream (see page 901)

23.1.11.4.7 TSOAPAttachment.SaveToStream Method

Saves the contents of the attachment to a specified file.

Pascal

```
procedure SaveToStream(AStream: TStream);
```

C++

```
__fastcall SaveToStream(TStream AStream);
```

Description

Call SaveToStream to write the contents of the attachment to a specified stream. This content can come from

The temporary file that the CacheFile (see page 904) property indicates.

The stream that the SourceStream (see page 907) property specifies.

The file that the SetSourceFile (see page 902) method supplies.

The string that the SourceString (see page 908) property specifies.

AStream is the stream to which SaveToStream writes the content of the attachment.

See Also

CacheFile (see page 904)

SourceStream (see page 907)

SourceString (see page 908)

SetSourceFile (see page 902)

23.1.11.4.8 TSOAPAttachment.SetOwnership Method

Pascal

```
procedure SetOwnership(const Value: TStreamOwnership); virtual;
```

C++

```
virtual __fastcall SetOwnership(const TStreamOwnership Value);
```

Description

This is SetOwnership, a member of class TSOAPAttachment.

23.1.11.4.9 TSOAPAttachment.SetSourceFile Method

Specifies a file from which TSOAPAttachment (see page 895) can read the content of the attachment.

Pascal

```
procedure SetSourceFile(const Value: string); virtual;
```

C++

```
virtual __fastcall SetSourceFile(const UnicodeString Value);
```

Description

Call SetSourceFile to supply the content of an attachment that the application sends from a file. SetSourceFile reads the content from the specified file, clears the SourceStream (see page 907) and SourceString (see page 908) properties, and sets the ContentType (see page 905) property to 'Application/binary'. If another MIME type string better describes the data, reset the ContentType (see page 905) property after calling SetSourceFile.

Value is the fully-qualified file name for the file whose content should be sent as an attachment.

See Also

ContentType (see page 905)

SetSourceStream (see page 903)

SourceString (see page 908)

23.1.11.4.10 TSOAPAttachment.SetSourceStream Method

Specifies a stream from which TSOAPAttachment (see page 895) can read the content of the attachment.

Pascal

```
procedure SetSourceStream(const Value: TStream; const Ownership: TStreamOwnership =
soReference); virtual;
```

C++

```
virtual __fastcall SetSourceStream(const TStream Value, const TStreamOwnership Ownership =
soReference);
```

Description

Call SetSourceStream to supply the content of an attachment that the application sends from a stream object. SetSourceStream assigns the specified stream as the value of the SourceStream (see page 907) property, clears the FileName and SourceString (see page 908) properties, and sets the ContentType (see page 905) property to 'Application/binary'. If another MIME type string better describes the data, reset the ContentType (see page 905) property after calling SetSourceStream.

Value is a stream from which TSOAPAttachment (see page 895) can read the content of the attachment.

Ownership (see page 907) indicates whether the TSOAPAttachment (see page 895) object should free the stream (Value) when it no longer needs it. It becomes the value of the Ownership (see page 907) property.

See Also

SourceStream (see page 907)

ContentType (see page 905)

Ownership (see page 907)

SourceString (see page 908)

SetSourceFile (see page 902)

23.1.11.4.11 TSOAPAttachment.SetSourceString Method

Pascal

```
procedure SetSourceString(const Value: string); virtual;
```

C++

```
virtual __fastcall SetSourceString(const UnicodeString Value);
```

Description

This is SetSourceString, a member of class TSOAPAttachment.

23.1.11.4.12 TSOAPAttachment.SOAPToObject Method

Initializes this TSOAPAttachment (see page 895) instance to represent the value of its SOAP representation.

Pascal

```
procedure SOAPToObject(const RootNode: IXMLNode; const Node: IXMLNode; const ObjConverter:
IObjConverter); override;
```

C++

```
virtual __fastcall SOAPToObject(const IXMLNode RootNode, const IXMLNode Node, const
IObjConverter ObjConverter);
```

Description

SOAPToObject initializes the published properties of this object instance so that they reflect the header's SOAP representation. In TSOAPAttachment (see page 895), this method calls the converter's FindAttachment method to locate information about the attachment, and then calls the Init (see page 899) method to initialize this object from the located information.

RootNode is an ancestor node of the attachment node that can contain information about the attachment node.

Node is the node in the SOAP envelope that represents a reference to the attachment.

ObjConverter is the interface of the default converter that translates between remotable objects and their SOAP representation. This is the interface to a TOPToSoapDomConvert instance.

In TSOAPAttachment (see page 895), SOAPToObject simply calls the InitObjectFromSOAP method of the converter supplied by the ObjConverter parameter.

See Also

ObjectToSOAP (see page 900)

FindAttachment (see page 922)

Init (see page 899)

23.1.11.5 TSOAPAttachment Properties

The properties of the TSOAPAttachment class are listed here.

Public Properties

	Name	Description
R	CacheFile (see page 904)	Identifies the temporary file where the attachment is saved.
	CacheFilePersist (see page 905)	Indicates whether the temporary file that contains the attachment data is deleted when TSOAPAttachment (see page 895) is freed.
	ContentType (see page 905)	Specifies the MIME type of the attachment data.
	Encoding (see page 906)	Indicates the encoding (character set) of the body of the attachment.
R	Headers (see page 906)	Indicates the values of the headers received with the attachment.
	Ownership (see page 907)	Indicates whether the TSOAPAttachment (see page 895) object should free its source stream.
R	SourceStream (see page 907)	Indicates a stream from which to read the content of an attachment that the application sends.
	SourceString (see page 908)	Specifies a string that is sent as the value of the attachment.

23.1.11.5.1 TSOAPAttachment.CacheFile Property

Identifies the temporary file where the attachment is saved.

Pascal

```
property CacheFile: string;
```

C++

```
__property UnicodeString CacheFile;
```

Description

CacheFile is set automatically when the application receives an attachment along with a SOAP message. This is the name of the temporary file where the application stores the contents of the attachment.

To save this temporary file to another file of your choosing, use the SaveToFile (see page 901) method. You can also save its contents to a stream, using the SaveToStream (see page 901) method.

The ContentType (see page 905) property indicates the format of the data in this temporary file.

By default, when the TSOAPAttachment (see page 895) object is freed, it deletes the temporary file that CacheFile indicates. Use the CacheFilePersist (see page 905) property to prevent this file from being deleted.

See Also

CacheFilePersist (see page 905)

ContentType (see page 905)

SaveToFile (see page 901)

SaveToStream (see page 901)

23.1.11.5.2 TSOAPAttachment.CacheFilePersist Property

Indicates whether the temporary file that contains the attachment data is deleted when TSOAPAttachment (see page 895) is freed.

Pascal

```
property CacheFilePersist: Boolean;
```

C++

```
__property Boolean CacheFilePersist;
```

Description

When CacheFilePersist is false (the default), the temporary file that the CacheFile (see page 904) property identifies is automatically deleted when the TSOAPAttachment (see page 895) instance is freed. You can save the information from the cache file using the SaveToFile (see page 901) or SaveToStream (see page 901) method before freeing the TSOAPAttachment (see page 895) instance.

When CacheFilePersist is true, the temporary file is not deleted when the TSOAPAttachment (see page 895) instance is freed.

See Also

CacheFile (see page 904)

SaveToFile (see page 901)

SaveToStream (see page 901)

23.1.11.5.3 TSOAPAttachment.ContentType Property

Specifies the MIME type of the attachment data.

Pascal

```
property ContentType: string;
```

C++

```
__property UnicodeString ContentType;
```

Description

Use ContentType to determine the MIME type (format) used for the data in the attachment.

When receiving an attachment, the ContentType property is automatically set to reflect the value of the content-type header of the part of the multipart form that holds the attachment. You can read this property to determine how to interpret the contents of the file that the CacheFile (see page 904) property indicates.

When sending an attachment, set ContentType to indicate the MIME type for the data you are sending. This property should not be set until after setting the SourceString (see page 908) property calling the SetSourceFile (see page 902) or SetSourceStream (see page 903) method. That is because calling SetSourceStream (see page 903) or SetSourceFile (see page 902) automatically sets ContentType to 'Application/binary', and setting SourceString (see page 908) automatically sets ContentType to 'Text/simple'. There is no need to reset ContentType if these values are correct.

See Also

CacheFile (see page 904)

SourceString (see page 908)

SetSourceFile (see page 902)

SetSourceStream (see page 903)

23.1.11.5.4 TSOAPAttachment.Encoding Property

Indicates the encoding (character set) of the body of the attachment.

Pascal

```
property Encoding: InvString;
```

C++

```
__property InvString Encoding;
```

Description

Use Encoding to look up or set the encoding type of the attachment's content. When sending an attachment, set this property to indicate the encoding that was used for the source you supply.

If you supply the attachment data as a string by setting the SourceString (see page 908) property, TSOAPAttachment (see page 895) assumes an encoding of UTF-8 if you do not set the Encoding property. If you supply the attachment data by calling SetSourceFile (see page 902) or SetSourceStream (see page 903), you must set the Encoding property, as TSOAPAttachment (see page 895) does not have a meaningful default encoding.

Note: Encoding indicates the encoding of the data that the SOAP attachment contains. The application does not automatically use this information to encode or decode that data.

See Also

ContentType (see page 905)

23.1.11.5.5 TSOAPAttachment.Headers Property

Indicates the values of the headers received with the attachment.

Pascal

```
property Headers: TStrings;
```

C++

```
__property TStrings Headers;
```

Description

Read Headers to determine the headers that appeared in the part of the multipart form that contained the attachment. Headers is only intended for attachments that the application receives. When sending attachments, the application automatically generates the headers that accompany the attachment.

Headers is a string list where each string has the form HeaderName=Value. HeaderName is the name of the header, such as

Content-Id, Content-Location, Content-Size, and so on. Value is the string that is the header value. Note that if you are only interested in the content type, you can use the ContentType (see page 905) property instead.

Note: Although the MIME specification states that the Content-Length header is optional, TSOAPAttachment (see page 895) requires this header and raises an exception if it is not present.

See Also

ContentType (see page 905)

CacheFile (see page 904)

23.1.11.5.6 TSOAPAttachment.Ownership Property

Indicates whether the TSOAPAttachment (see page 895) object should free its source stream.

Pascal

```
property Ownership: TStreamOwnership;
```

C++

```
__property TStreamOwnership Ownership;
```

Description

Ownership indicates whether TSOAPAttachment (see page 895) needs to free the stream that supplied the content of the attachment when the application supplied this data using the SetSourceStream (see page 903) method.

See Also

SourceStream (see page 907)

SetSourceStream (see page 903)

23.1.11.5.7 TSOAPAttachment.SourceStream Property

Indicates a stream from which to read the content of an attachment that the application sends.

Pascal

```
property SourceStream: TStream;
```

C++

```
__property TStream SourceStream;
```

Description

Read SourceStream to access the content of an attachment that the application is sending. When SourceStream is not nil (Delphi) or NULL (C++), the content of the attachment is read from this stream and the SourceString (see page 908) property is not set. Depending on the value of the Ownership (see page 907) property, SourceStream can be owned by the TSOAPAttachment (see page 895) object (in which case TSOAPAttachment (see page 895) frees the stream object when it is no longer needed) or can exist as an independent object that TSOAPAttachment (see page 895) does not free.

SourceStream is a read-only property. To specify a stream from which TSOAPAttachment (see page 895) reads its data, use the SetSourceStream (see page 903) method.

See Also

SourceString (see page 908)

SetSourceStream (see page 903)

Ownership (see page 907)

ContentType (see page 905)

23.1.11.5.8 TSOAPAttachment.SourceString Property

Specifies a string that is sent as the value of the attachment.

Pascal

```
property SourceString: string;
```

C++

```
__property UnicodeString SourceString;
```

Description

Use SourceString to supply the content of an attachment as a string.

Setting the SourceString property causes TSOAPAttachment (see page 895) to clear the SourceStream (see page 907) property and to set the ContentType (see page 905) property to 'Text/simple'. If another MIME type string better describes the data, reset the ContentType (see page 905) property after setting SourceString.

See Also

SourceStream (see page 907)

SetSourceStream (see page 903)

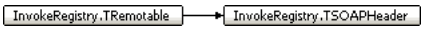
ContentType (see page 905)

SetSourceFile (see page 902)

23.1.12 TSOAPHeader Class

TSOAPHeader represents a header on the SOAP envelope that contains an encoded message in a Web Service application.

Class Hierarchy



Pascal

```
TSOAPHeader = class(TRemotable);
```

C++

```
class TSOAPHeader : public TRemotable;
```

File

InvokeRegistry

Description

TSOAPHeader is the base class for classes that represent SOAP headers. Each type of header has an associated TSOAPHeader descendant, which is registered with the remotable type registry. Register a TSOAPHeader descendant like any other remotable type, using the name of the header as the type name. If you import a WSDL document, the code to define and register the TSOAPHeader descendant is generated automtically.

You can also register the TSOAPHeader descendant with the invocation registry. The invocation registry's RegisterHeaderClass method associates the header class and the invokable interfaces with which it should be included. Registering the header with the invocation registry serves two purposes:

When a server publishes WSDL information about the Web Service, information about the header type is included in the resulting WSDL document.

On both servers and clients, the invocation registry associates the header class with a header name and a namespace in which that name is defined. This is important if the namespace for the header type is different than the namespace for the invokable interface that uses the header.

A SOAP header is a special remotable type that represents a node in the header section of a SOAP envelope. As with any other remotable type, the published properties of the TSOAPHeader descendant correspond to the child nodes and attributes of the corresponding header node. TSOAPHeader introduces two properties for common attributes of SOAP headers: MustUnderstand (see page 912) and Actor (see page 911). Typically, SOAP headers represent information that is associated with an entire invokable interface (port type) rather than just the method that is the focus of a single message.

When an application receives a SOAP message (that is, when a server receives a request to execute a method on an invokable interface or when a client receives a response after sending such a request), the headers on that message are automatically converted to the corresponding TSOAPHeader descendants that are registered with the remotable type registry. You can access these TSOAPHeader descendants using the ISOAPHeaders (see page 923) interface, which can be obtained from the implementation class of the invokable interface or from the invokable interface of a remote interfaced object.

When an application sends a SOAP message (either a client sending a request or a server sending a response), supply the headers that should appear on that outgoing message by creating TSOAPHeader descendants and calling the Send method of an ISOAPHeaders (see page 923) interface.

See Also

ISOAPHeaders (see page 923)

TRemotableTypeRegistry (see page 874)

TInvokableClassRegistry (see page 840)

23.1.12.1 TSOAPHeader Members

The following tables list the members exposed by TSOAPHeader.

Protected Fields

	Name	Description
	FSerializationOptions (see page 871)	This is FSerializationOptions, a member of class TRemotable.

Public Methods

	Name	Description
	Create (see page 870)	Creates and initializes a TRemotable (see page 868) object.
	Destroy (see page 870)	Frees the memory associated with the TRemotable (see page 868) object.
	ObjectToSOAP (see page 910)	Converts this TSOAPHeader (see page 908) object to its SOAP representation.
	SOAPToObject (see page 911)	Initializes this TSOAPHeader (see page 908) instance to represent the value of its SOAP representation.

Public Properties

	Name	Description
	Actor (see page 911)	Identifies the agent to which the header is directed.
	DataContext (see page 873)	Specifies the object that controls the lifetime of a TRemotable (see page 868) instance.
	MustUnderstand (see page 912)	Indicates whether an application is required to recognize and handle the header on received messages.
	SerializationOptions (see page 873)	Indicates how the value of the TRemotable (see page 868) descendant encodes its value.

23.1.12.2 TSOAPHeader Methods

The methods of the TSOAPHeader class are listed here.

Public Methods

	Name	Description
	ObjectToSOAP (see page 910)	Converts this TSOAPHeader (see page 908) object to its SOAP representation.
	SOAPToObject (see page 911)	Initializes this TSOAPHeader (see page 908) instance to represent the value of its SOAP representation.

23.1.12.2.1 TSOAPHeader.ObjectToSOAP Method

Converts this TSOAPHeader (see page 908) object to its SOAP representation.

Pascal

```
function ObjectToSOAP(RootNode: IXMLNode; ParentNode: IXMLNode; const ObjConverter:
IObjConverter; const Name: InvString; const URI: InvString; ObjConvOpts:
TObjectConvertOptions; out RefID: InvString): IXMLNode; override;
```

C++

```
virtual __fastcall IXMLNode ObjectToSOAP(IXMLNode RootNode, IXMLNode ParentNode, const
IObjConverter ObjConverter, const InvString Name, const InvString URI, TObjectConvertOptions
ObjConvOpts, InvString RefID);
```

Description

Override ObjectToSOAP to customize the way your application converts a TSOAPHeader (see page 908) descendant to its SOAP representation. In TSOAPHeader (see page 908), ObjectToSOAP uses the default conversion (supplied by the ObjConverter parameter), adding a MustUnderstand (see page 912) attribute whose value reflects the value of the MustUnderstand (see page 912) property.

RootNode is an ancestor of the generated node to which information about the generated node can be added as attributes.

ParentNode is the node in the SOAP representation that should act as a parent to the node that ObjectToSOAP generates.

ObjConverter is the interface of the default converter that translates between remotable objects and their SOAP representation. This is the interface to a TOPToSoapDomConvert instance.

Name is the name of the node that is generated.

URI is the URI for the namespace in which Name is defined.

ObjConvOpts are flags that customize the way the node is serialized.

RefID returns a reference ID for the node. This is used if the node is multi-referenced.

ObjectToSOAP returns the XML for the newly generated node.

See Also

MustUnderstand (see page 912)

ObjInstanceToSOAP (see page 922)

SOAPToObject (see page 911)

23.1.12.2.2 TSOAPHeader.SOAPToObject Method

Initializes this TSOAPHeader (see page 908) instance to represent the value of its SOAP representation.

Pascal

```
procedure SOAPToObject(const RootNode: IXMLNode; const Node: IXMLNode; const ObjConverter:
IObjConverter); override;
```

C++

```
virtual __fastcall SOAPToObject(const IXMLNode RootNode, const IXMLNode Node, const
IObjConverter ObjConverter);
```

Description

SOAPToObject initializes the published properties of this object instance so that they reflect the header's SOAP representation. Override this method to customize the way your application creates a specific type of header from its XML representation.

RootNode is an ancestor node of the header node that can contain information about the header node.

Node is the node in the SOAP envelope that represents the header.

ObjConverter is the interface of the default converter that translates between remotable objects and their SOAP representation. This is the interface to a TOPToSoapDomConvert instance.

See Also

ObjectToSOAP (see page 910)

InitObjectFromSOAP (see page 922)

MustUnderstand (see page 912)

23.1.12.3 TSOAPHeader Properties

The properties of the TSOAPHeader class are listed here.

Public Properties

	Name	Description
	Actor (see page 911)	Identifies the agent to which the header is directed.
	MustUnderstand (see page 912)	Indicates whether an application is required to recognize and handle the header on received messages.

23.1.12.3.1 TSOAPHeader.Actor Property

Identifies the agent to which the header is directed.

Pascal

```
property Actor: InvString;
```

C++

```
__property InvString Actor;
```

Description

Actor is the URI to which the header is directed. Set Actor to indicate the ultimate target of a header you are sending. Read Actor to determine whether a header the application receives is supposed to be handled by the application rather than merely passed on to the next processor.

Note: When the Actor property of a received header identifies this application and the application raises an exception concerning that header (such as a MustUnderstand (see page 912) fault for a header that it does not know how to handle), the value of Actor should become the value of the FaultActor property on the ERemotableException (see page 810) object.

See Also

MustUnderstand (see page 912)

FaultActor (see page 812)

23.1.12.3.2 TSOAPHeader.MustUnderstand Property

Indicates whether an application is required to recognize and handle the header on received messages.

Pascal

```
property MustUnderstand: Boolean;
```

C++

```
__property Boolean MustUnderstand;
```

Description

When MustUnderstand is false, applications can ignore this header if they do not recognize it and know how to handle it.

When MustUnderstand is true, applications that receive messages with this header but that do not recognize or handle it must throw a MustUnderstand fault.

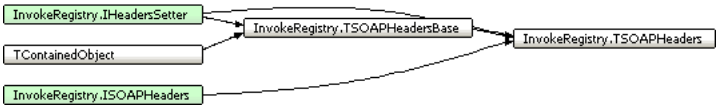
See Also

Actor (see page 911)

23.1.13 TSOAPHeaders Class

TSOAPHeaders is a helper object for implementing the ISOAPHeaders (see page 923) interface.

Class Hierarchy



Pascal

```
TSOAPHeaders = class(TSOAPHeadersBase, ISOAPHeaders, IHeadersSetter);
```

C++

```
class TSOAPHeaders : public TSOAPHeadersBase, public ISOAPHeaders, public IHeadersSetter;
```

File

InvokeRegistry

Description

TSOAPHeaders implements the ISOAPHeaders (see page 923) interface for a class that needs to provide access to the headers of SOAP messages that the application sends and receives Typically, an application does not work directly with instances of TSOAPHeaders. Rather, it uses the ISOAPHeaders (see page 923) interface that this object implements for another class, such as TRIO or TInvokableClass (see page 833).

See Also

ISOAPHeaders (see page 923)

TSOAPHeader (see page 908)

TRIO (see page 4056)

TInvokableClass (see page 833)

23.1.13.1 TSOAPHeaders Members

The following tables list the members exposed by TSOAPHeaders.

Protected Fields

	Name	Description
	FHeadersInbound (see page 917)	This is FHeadersInbound, a member of class TSOAPHeadersBase.
	FHeadersOutBound (see page 918)	This is FHeadersOutBound, a member of class TSOAPHeadersBase.

Protected Methods

	Name	Description
	SetHeadersInOut (see page 918)	Sets the lists of incoming and outgoing headers.

Public Methods

	Name	Description
	Get (see page 924)	This is the overview for the Get method overload.
	Get (see page 914)	This is the overview for the Get method overload.
	GetOwnsSentHeaders (see page 914)	Indicates whether the TSOAPHeaders (see page 912) object frees headers that are added using the Send (see page 915) method
	Send (see page 915)	Adds a header to an outgoing message.
	SendAt (see page 915)	Retrieves a header from an outgoing message, given its position.
	SendCount (see page 916)	Indicates the number of headers that have been added to an outgoing message.
	SetHeadersInOut (see page 920)	This is SetHeadersInOut, a member of class IHeadersSetter.
	SetOwnsSentHeaders (see page 916)	Sets a flag that indicates who should free outgoing header objects.

Public Properties

	Name	Description
	OwnsSentHeaders (see page 927)	Indicates who is responsible for freeing header objects that are included in outbound messages.

23.1.13.2 TSOAPHeaders Methods

The methods of the TSOAPHeaders class are listed here.

Public Methods

	Name	Description
	Get (see page 914)	This is the overview for the Get method overload.
	GetOwnsSentHeaders (see page 914)	Indicates whether the TSOAPHeaders (see page 912) object frees headers that are added using the Send (see page 915) method
	Send (see page 915)	Adds a header to an outgoing message.
	SendAt (see page 915)	Retrieves a header from an outgoing message, given its position.
	SendCount (see page 916)	Indicates the number of headers that have been added to an outgoing message.
	SetOwnsSentHeaders (see page 916)	Sets a flag that indicates who should free outgoing header objects.

23.1.13.2.1 Get Method

This is the overview for the Get method overload.

Overload List

	Name	Description
	TSOAPHeaders.Get (InvString, InvString) (see page 914)	Retrieves a header from an incoming message.
	TSOAPHeaders.Get (TClass) (see page 914)	Retrieves a header from an incoming message.
	TSOAPHeaders.Get (TClass, TSOAPHeader) (see page 914)	Retrieves a header from an incoming message.

23.1.13.2.1.1 TSOAPHeaders.Get Method (InvString, InvString)

Retrieves a header from an incoming message.

Pascal

```
procedure Get(Cls: TClass; out Hdr: TSOAPHeader); overload;
function Get(Cls: TClass): TSOAPHeader; overload;
function Get(const Name: InvString; const URI: InvString): TSOAPHeader; overload;
```

C++

```
__fastcall Get(TClass Cls, TSOAPHeader Hdr);
__fastcall TSOAPHeader Get(TClass Cls);
__fastcall TSOAPHeader Get(const InvString Name, const InvString URI);
```

Description

Call Get to access the header object that is automatically created to represent one of the headers on an incoming messages. On the server, this is a header on the request message. On the client, this is a header on the response message.

Cls is the class reference that indicates the type of header class you want to retrieve. Each header class corresponds to a type of header, where the type name under which the class is registered corresponds to the name of the header node.

Hdr returns a TSOAPHeader (see page 908) descendant that represents the header of the incoming message. If this syntax is not used, Get returns the header instance.

Name is the name of the header (the node name for the header node) that you want to retrieve.

URI is the URI for the namespace in which Name is defined.

See Also

Send (see page 915)

DataContext (see page 873)

23.1.13.2.2 TSOAPHeaders.GetOwnsSentHeaders Method

Indicates whether the TSOAPHeaders (see page 912) object frees headers that are added using the Send (see page 915) method

Pascal

```
function GetOwnsSentHeaders: Boolean;
```

C++

```
__fastcall Boolean GetOwnsSentHeaders();
```

Description

GetOwnsSentHeaders returns true if the TSOAPHeaders (see page 912) instance frees any header objects added using the Send (see page 915) method. It returns false if the caller is responsible for freeing these header objects.

See Also

Send (see page 915)

23.1.13.2.3 TSOAPHeaders.Send Method

Adds a header to an outgoing message.

Pascal

```
procedure Send(const Hdr: TSOAPHeader);
```

C++

```
__fastcall Send(const TSOAPHeader Hdr);
```

Description

Call Send to add a header to an outgoing message. On the client, call Send before executing a method on an invokable interface. On the server, call Send before returning from the method that the client invoked.

Hdr is the header object instance that represents the header you want to add to the outgoing message.

Note: The SetOwnsSentHeaders (see page 916) method controls whether you must free the object that is supplied as the Hdr parameter after sending the message.

See Also

SendAt (see page 915)

SendCount (see page 916)

SetOwnsSentHeaders (see page 916)

23.1.13.2.4 TSOAPHeaders.SendAt Method

Retrieves a header from an outgoing message, given its position.

Pascal

```
function SendAt(Index: Integer): TSOAPHeader;
```

C++

```
__fastcall TSOAPHeader SendAt(int Index);
```

Description

Call SendAt to retrieve the header object for a header you have already added to an outgoing message using the Send (see page 915) method.

Index is the position of the header you want to retrieve, where 0 specifies the first header, 1 specifies the second header, and so on. The SendCount (see page 916) property provides an upper bound for this index.

SendAt returns the header at the specified position.

See Also

Send (see page 915)

SendCount (see page 916)

23.1.13.2.5 TSOAPHeaders.SendCount Method

Indicates the number of headers that have been added to an outgoing message.

Pascal

```
function SendCount: Integer;
```

C++

```
__fastcall int SendCount();
```

Description

Call SendCount to determine the number of headers that have been added to an outgoing message. These headers are added by previous calls to the Send (see page 915) method.

See Also

Send (see page 915)

SendAt (see page 915)

23.1.13.2.6 TSOAPHeaders.SetOwnsSentHeaders Method

Sets a flag that indicates who should free outgoing header objects.

Pascal

```
procedure SetOwnsSentHeaders(Flag: Boolean);
```

C++

```
__fastcall SetOwnsSentHeaders(Boolean Flag);
```

Description

Call SetOwnsSentHeaders to specify who should free header objects that are added using the Send (see page 915) method.

Flag indicates whether this TSOAPHeaders (see page 912) frees the outgoing headers. If Flag is true, TSOAPHeaders (see page 912) frees the headers that are added by calling Send (see page 915). If Flag is false, the caller is responsible for freeing these header objects.

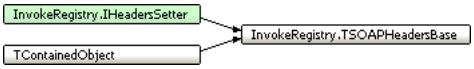
See Also

Send (see page 915)

23.1.14 TSOAPHeadersBase Class

TSOAPHeadersBase is the base class for classes that maintain lists of SOAP headers.

Class Hierarchy



Pascal

```
TSOAPHeadersBase = class(TContainedObject, IHeadersSetter);
```

C++

```
class TSOAPHeadersBase : public TContainedObject, public IHeadersSetter;
```

File

InvokeRegistry

Description

TSOAPHeadersBase provides support for maintaining two lists of TSOAPHeader (see page 908) descendants. One list is for the headers that an application sends and the other is for the headers that are it receives with an incoming message.

TSOAPHeadersBase is a descendant of TContainedObject. As such, it is designed to be used as a helper object in supporting an interface for working with SOAP headers. TSOAPHeadersBase is intended only as a base class. For working with SOAP headers, use a TSOAPHeadersBase descendant such as TSOAPHeaders (see page 912).

See Also

TSOAPHeaders (see page 912)

TSOAPHeader (see page 908)

23.1.14.1 TSOAPHeadersBase Members

The following tables list the members exposed by TSOAPHeadersBase.

Protected Fields

	Name	Description
	FHeadersInbound (see page 917)	This is FHeadersInbound, a member of class TSOAPHeadersBase.
	FHeadersOutBound (see page 918)	This is FHeadersOutBound, a member of class TSOAPHeadersBase.

Protected Methods

	Name	Description
	SetHeadersInOut (see page 918)	Sets the lists of incoming and outgoing headers.

23.1.14.2 TSOAPHeadersBase Fields

The fields of the TSOAPHeadersBase class are listed here.

Protected Fields

	Name	Description
	FHeadersInbound (see page 917)	This is FHeadersInbound, a member of class TSOAPHeadersBase.
	FHeadersOutBound (see page 918)	This is FHeadersOutBound, a member of class TSOAPHeadersBase.

23.1.14.2.1 TSOAPHeadersBase.FHeadersInbound Field

Pascal

```
FHeadersInbound: THeaderList;
```

C++

```
THeaderList FHeadersInbound;
```

Description

This is FHeadersInbound, a member of class TSOAPHeadersBase.

23.1.14.2.2 TSOAPHeadersBase.FHeadersOutBound Field

Pascal

```
FHeadersOutBound: THeaderList;
```

C++

```
THeaderList FHeadersOutBound;
```

Description

This is FHeadersOutBound, a member of class TSOAPHeadersBase.

23.1.14.3 TSOAPHeadersBase Methods

The methods of the TSOAPHeadersBase class are listed here.

Protected Methods

	Name	Description
	SetHeadersInOut (see page 918)	Sets the lists of incoming and outgoing headers.

23.1.14.3.1 TSOAPHeadersBase.SetHeadersInOut Method

Sets the lists of incoming and outgoing headers.

Pascal

```
procedure SetHeadersInOut(var InHdrs: THeaderList; var OutHdrs: THeaderList);
```

C++

```
__fastcall SetHeadersInOut(THeaderList InHdrs, THeaderList OutHdrs);
```

Description

Applications can't call the protected SetHeadersInOut method. It is used by descendant classes to initialize the lists of incoming and outgoing header objects.

InHdrs is a list of TSOAPHeader (see page 908) descendants that represent the headers received in a SOAP message.

OutHdrs is a list of TSOAPHeader (see page 908) descendants that represent the headers sent in a SOAP message.

See Also

TSOAPHeader (see page 908)

23.2 Interfaces

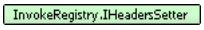
The following table lists interfaces in this documentation.

Interfaces

	Name	Description
	IHeadersSetter (see page 919)	This is class InvokeRegistry.IHeadersSetter.
	IObjConverter (see page 920)	IObjConverter is the interface for converting between a remotable class and its SOAP representation.
	ISOAPHeaders (see page 923)	ISOAPHeaders is the interface for accessing the headers that accompany a message encoded using SOAP.

23.2.1 IHeadersSetter Interface

Class Hierarchy



Pascal

```
IHeadersSetter = interface;
```

C++

```
__interface IHeadersSetter;
```

File

InvokeRegistry

Description

This is class InvokeRegistry.IHeadersSetter.

23.2.1.1 IHeadersSetter Members

The following tables list the members exposed by IHeadersSetter.

Public Methods

	Name	Description
	SetHeadersInOut (see page 920)	This is SetHeadersInOut, a member of class IHeadersSetter.

23.2.1.2 IHeadersSetter Methods

The methods of the IHeadersSetter class are listed here.

Public Methods

	Name	Description
	SetHeadersInOut (see page 920)	This is SetHeadersInOut, a member of class IHeadersSetter.

23.2.1.2.1 IHeadersSetter.SetHeadersInOut Method

Pascal

```
procedure SetHeadersInOut(var InHdrs: THeaderList; var OutHdrs: THeaderList);
```

C++

```
__fastcall SetHeadersInOut(THeaderList InHdrs, THeaderList OutHdrs);
```

Description

This is SetHeadersInOut, a member of class IHeadersSetter.

23.2.2 IObjConverter Interface

IObjConverter is the interface for converting between a remotable class and its SOAP representation.

Class Hierarchy

Pascal

```
IObjConverter = interface;
```

C++

```
__interface IObjConverter;
```

File

InvokeRegistry

Description

IObjConverter defines methods that aid in the conversion between remotable classes (classes that descend from TRemotable (see page 868)) and their representation in the messages that pass between clients and Web Service providers. That is, this interface provides methods that are used when converting between a remotable class, whether it represents a complex type in an invokable interface, an attachment, or a SOAP header, and its SOAP representation. The TSOAPDomConv class implements this interface.

Note: C++ method declarations that use IObjConverter use the _di_IObjConverter type instead. This type is a DelphiInterface wrapper around the IObjConverter interface:

Note: typedef System::DelphiInterface< IObjConverter > _di_IObjConverter;

See Also

TSOAPDomConv (see page 2758)

TRemotable (see page 868)

TSOAPHeader (see page 908)

TSOAPAttachment (see page 895)

23.2.2.1 IObjConverter Members

The following tables list the members exposed by IObjConverter.

Public Methods

	Name	Description
	AddAttachment (see page 921)	Stores information about a SOAP attachment so that it is included in the final message that is created.
	FindAttachment (see page 922)	Returns a TSOAPAttachment (see page 895) instance that represent the attachment with a specified ID.
	InitObjectFromSOAP (see page 922)	Initializes an instance of a remotable object to reflect the values in its SOAP representation.
	ObjInstanceToSOAP (see page 922)	Returns the SOAP representation of a remotable object.

23.2.2.2 IObjConverter Methods

The methods of the IObjConverter class are listed here.

Public Methods

	Name	Description
	AddAttachment (see page 921)	Stores information about a SOAP attachment so that it is included in the final message that is created.
	FindAttachment (see page 922)	Returns a TSOAPAttachment (see page 895) instance that represent the attachment with a specified ID.
	InitObjectFromSOAP (see page 922)	Initializes an instance of a remotable object to reflect the values in its SOAP representation.
	ObjInstanceToSOAP (see page 922)	Returns the SOAP representation of a remotable object.

23.2.2.2.1 IObjConverter.AddAttachment Method

Stores information about a SOAP attachment so that it is included in the final message that is created.

Pascal

```
procedure AddAttachment(Attachment: TSOAPAttachment; const AContentId: string);
```

C++

```
__fastcall AddAttachment(TSOAPAttachment Attachment, const UnicodeString AContentId);
```

Description

AddAttachment stores a SOAP attachment so that it can be used to generate part of the multipart form that includes the SOAP envelope as another of its parts.

Attachment is the attachment that should be added to the multipart form.

AContentId is a string that uniquely identifies the attachment. It is added as a header to the part of the form that contains the attachment so that XML in the SOAP envelope can refer to the attachment. When interpreting the encoded SOAP envelope, this value is passed to the FindAttachment (see page 922) method so that the information in the attachment part can be used to recreate the TSOAPAttachment (see page 895) instance.

See Also

FindAttachment (see page 922)

23.2.2.2.2 IObjConverter.FindAttachment Method

Returns a TSOAPAttachment (see page 895) instance that represent the attachment with a specified ID.

Pascal

```
function FindAttachment(const AContentId: string): TSOAPAttachment;
```

C++

```
__fastcall TSOAPAttachment FindAttachment(const UnicodeString AContentId);
```

Description

FindAttachment locates the attachment that is identified by a specified content ID (HREF).

AContentId is the unique identifier for the attachment that was supplied to the AddAttachment (see page 921) method when the attachment was first added to the multipart form.

FindAttachment returns a TSOAPAttachment (see page 895) instance that represents the attachment.

See Also

AddAttachment (see page 921)

23.2.2.2.3 IObjConverter.InitObjectFromSOAP Method

Initializes an instance of a remotable object to reflect the values in its SOAP representation.

Pascal

```
procedure InitObjectFromSOAP(Instance: TObject; RootNode: IXMLNode; Node: IXMLNode);
```

C++

```
__fastcall InitObjectFromSOAP(TObject * Instance, IXMLNode RootNode, IXMLNode Node);
```

Description

InitObjectFromSOAP assigns values to the published properties of a remotable object that represents a node from the SOAP envelope.

Instance is the remotable object to initialize from its SOAP representation.

RootNode is an ancestor node of the node for the remotable object that can contain information needed for initializing the remotable object.

Node is the node in the SOAP envelope that represents the remotable object.

See Also

ObjInstanceToSOAP (see page 922)

23.2.2.2.4 IObjConverter.ObjInstanceToSOAP Method

Returns the SOAP representation of a remotable object.

Pascal

```
function ObjInstanceToSOAP(Instance: TObject; RootNode: IXMLNode; ParentNode: IXMLNode; const
NodeName: InvString; const NodeNamespace: InvString; ObjConvOpts: TObjectConvertOptions; out
RefID: InvString): IXMLNode;
```

C++

```
__fastcall IXMLNode ObjInstanceToSOAP(TObject * Instance, IXMLNode RootNode, IXMLNode
```

```
ParentNode, const InvString NodeName, const InvString NodeNamespace, TObjectConvertOptions
ObjConvOpts, InvString RefID);
```

Description

ObjInstanceToSOAP adds a node to represent the value of a remotable object.

Instance is the remotable object to encode as its SOAP representation.

RootNode is an ancestor of the node to which information about the node can be added as attributes.

ParentNode is the node in the SOAP representation that should act as a parent to the node that ObjInstanceToSOAP generates.

NodeName is the name of the node that is generated.

NodeNamespace is the URI for the namespace in which NodeName is defined.

ObjConvOpts are flags that customize the way the node is serialized.

RefID returns a reference ID for the node. This is used if the node is multi-referenced.

ObjInstanceToSOAP returns the XML for the newly generated node.

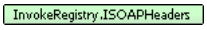
See Also

InitObjectFromSOAP (see page 922)

23.2.3 ISOAPHeaders Interface

ISOAPHeaders is the interface for accessing the headers that accompany a message encoded using SOAP.

Class Hierarchy



Pascal

```
ISOAPHeaders = interface;
```

C++

```
__interface ISOAPHeaders;
```

File

InvokeRegistry

Description

ISOAPHeaders is the interface for accessing the headers of a SOAP envelope. It defines methods for adding headers that are sent with an outgoing message and for retrieving the headers that were included in an incoming message.

On the server, the implementation class for in invokable interface supports this interface. You can obtain a reference to the ISOAPHeaders interface from the implementation class (a TInvokableClass (see page 833) descendant). You can then use this interface to retrieve the headers that were included in an incoming request or to add headers to an outgoing response message. If your implementation class does not descend from TInvokableClass (see page 833), you can obtain a reference to this interface using the global GetSOAPHeaders function.

For client-side header support, TRIO implements the ISOAPHeaders interface. However, you can't obtain an ISOAPHeaders interface directly from a remote interfaced object, because when you try to obtain an interface from TRIO or one of its descendants, it assumes that you are trying to obtain an invokable interface for a server object and generates an in-memory vtable. Instead, you must access the ISOAPHeaders interface from the invokable interface that the remote interface object supports.

Note: C++ method declarations that use ISOAPHeaders use the _di_ISOAPHeaders type instead. This type is a DelphiInterface wrapper around the ISOAPHeaders interface:

Note: typedef System::DelphiInterface< ISOAPHeaders > _di_ISOAPHeaders;

See Also

TRIO (see page 4056)

TInvokableClass (see page 833)

TSOAPHeaders (see page 912)

GetSOAPHeaders

Defining and Using SOAP Headers

Processing Headers in Client Applications

23.2.3.1 ISOAPHeaders Members

The following tables list the members exposed by ISOAPHeaders.

Public Methods

	Name	Description
	Get (see page 924)	This is the overview for the Get method overload.
	GetOwnsSentHeaders (see page 925)	Returns the value of the OwnsSentHeaders (see page 927) property.
	Send (see page 926)	Adds a header to an outgoing message.
	SendAt (see page 926)	Retrieves a header from an outgoing message, given its position.
	SendCount (see page 927)	Indicates the number of headers that have been added to an outgoing message.
	SetOwnsSentHeaders (see page 927)	Sets the value of the OwnsSentHeaders (see page 927) property.

Public Properties

	Name	Description
	OwnsSentHeaders (see page 927)	Indicates who is responsible for freeing header objects that are included in outbound messages.

23.2.3.2 ISOAPHeaders Methods

The methods of the ISOAPHeaders class are listed here.

Public Methods

	Name	Description
	Get (see page 924)	This is the overview for the Get method overload.
	GetOwnsSentHeaders (see page 925)	Returns the value of the OwnsSentHeaders (see page 927) property.
	Send (see page 926)	Adds a header to an outgoing message.
	SendAt (see page 926)	Retrieves a header from an outgoing message, given its position.
	SendCount (see page 927)	Indicates the number of headers that have been added to an outgoing message.
	SetOwnsSentHeaders (see page 927)	Sets the value of the OwnsSentHeaders (see page 927) property.

23.2.3.2.1 Get Method

This is the overview for the Get method overload.

Overload List

	Name	Description
	ISOAPHeaders.Get (InvString, InvString) (see page 925)	Retrieves a header from an incoming message.
	ISOAPHeaders.Get (TClass) (see page 925)	Retrieves a header from an incoming message.
	ISOAPHeaders.Get (TClass, TSOAPHeader) (see page 925)	Retrieves a header from an incoming message.

23.2.3.2.1.1 ISOAPHeaders.Get Method (InvString, InvString)

Retrieves a header from an incoming message.

Pascal

```
procedure Get(Cls: TClass; out Hdr: TSOAPHeader); overload;
function Get(Cls: TClass): TSOAPHeader; overload;
function Get(const Name: InvString; const URI: InvString): TSOAPHeader; overload;
```

C++

```
__fastcall Get(TClass Cls, TSOAPHeader Hdr);
__fastcall TSOAPHeader Get(TClass Cls);
__fastcall TSOAPHeader Get(const InvString Name, const InvString URI);
```

Description

Call Get to access the header object that is automatically created to represent one of the headers on an incoming messages. On the server, this is a header on the request message. On the client, this is a header on the response message.

Cls is the class reference that indicates the type of header class you want to retrieve. Each header class corresponds to a type of header, where the type name under which the class is registered corresponds to the name of the header node.

Hdr returns a TSOAPHeader (see page 908) descendant that represents the header of the incoming message. If this syntax is not used, Get returns the header instance.

Name is the name of the header (the node name for the header node) that you want to retrieve.

URI is the URI for the namespace in which Name is defined.

See Also

Send (see page 926)

DataContext (see page 873)

23.2.3.2.2 ISOAPHeaders.GetOwnsSentHeaders Method

Returns the value of the OwnsSentHeaders (see page 927) property.

Pascal

```
function GetOwnsSentHeaders: Boolean;
```

C++

```
__fastcall Boolean GetOwnsSentHeaders();
```

Description

GetOwnsSentHeaders is the read implementation for the OwnsSentHeaders (see page 927) property.

See Also

OwnsSentHeaders (see page 927)

23.2.3.2.3 ISOAPHeaders.Send Method

Adds a header to an outgoing message.

Pascal

```
procedure Send(const Hdr: TSOAPHeader);
```

C++

```
__fastcall Send(const TSOAPHeader Hdr);
```

Description

Call Send to add a header to an outgoing message. On the client, call Send before executing a method on an invokable interface. On the server, call Send before returning from the method that the client invoked.

Hdr is the header object instance that represents the header you want to add to the outgoing message.

Note: The OwnsSentHeaders (see page 927) property controls whether you must free the object that is supplied as the Hdr parameter after sending the message.

See Also

SendAt (see page 926)

SendCount (see page 927)

OwnsSentHeaders (see page 927)

23.2.3.2.4 ISOAPHeaders.SendAt Method

Retrieves a header from an outgoing message, given its position.

Pascal

```
function SendAt(Index: Integer): TSOAPHeader;
```

C++

```
__fastcall TSOAPHeader SendAt(int Index);
```

Description

Call SendAt to retrieve the header object for a header you have already added to an outgoing message using the Send (see page 926) method.

Index is the position of the header you want to retrieve, where 0 specifies the first header, 1 specifies the second header, and so on. The SendCount (see page 927) property provides an upper bound for this index.

SendAt returns the header at the specified position.

See Also

Send (see page 926)

SendCount (see page 927)

23.2.3.2.5 ISOAPHeaders.SendCount Method

Indicates the number of headers that have been added to an outgoing message.

Pascal

```
function SendCount: Integer;
```

C++

```
__fastcall int SendCount();
```

Description

Call SendCount to determine the number of headers that have been added to an outgoing message. These headers are added by previous calls to the Send (see page 926) method.

See Also

Send (see page 926)

SendAt (see page 926)

23.2.3.2.6 ISOAPHeaders.SetOwnsSentHeaders Method

Sets the value of the OwnsSentHeaders (see page 927) property.

Pascal

```
procedure SetOwnsSentHeaders(Flag: Boolean);
```

C++

```
__fastcall SetOwnsSentHeaders(Boolean Flag);
```

Description

SetOwnsSentHeaders is the write implementation for the OwnsSentHeaders (see page 927) property.

Flag is the new value of the OwnsSentHeaders (see page 927) property.

See Also

OwnsSentHeaders (see page 927)

23.2.3.3 ISOAPHeaders Properties

The properties of the ISOAPHeaders class are listed here.

Public Properties

	Name	Description
	OwnsSentHeaders (see page 927)	Indicates who is responsible for freeing header objects that are included in outbound messages.

23.2.3.3.1 ISOAPHeaders.OwnsSentHeaders Property

Indicates who is responsible for freeing header objects that are included in outbound messages.

Pascal

```
property OwnsSentHeaders: Boolean;
```

C++

```
__property Boolean OwnsSentHeaders;
```

Description

Use OwnsSentHeaders to indicate whether the header object instances you create and add to messages by calling the Send (see page 926) or SendAt (see page 926) method should be freed by the object that implements this interface. When OwnsSentHeaders is true, the object implementing ISOAPHeaders (see page 923) frees the header objects after sending them. When OwnsSentHeaders is false, the caller is responsible for freeing the header objects.

Note: For inbound headers (headers retrieved from incoming messages), the object that implements always frees them unless you call the Get (see page 925) method. After a call to Get (see page 925), the caller is responsible for freeing the header, except on the server, where the header object is freed by the header object's DataContext.

See Also

Send (see page 926)

SendAt (see page 926)

DataContext (see page 873)

23.3 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	GetRemotableDataContext (see page 929)	Returns the TDataContext (see page 814) object that provides temporary storage during a call to an invokable interface.
	InvRegistry (see page 930)	Provides access to the invocation registry.
	RemClassRegistry (see page 930)	Provides access to the remotable class registry.
	RemTypeRegistry (see page 931)	Provides access to the remotable type registry.
	SetRemotableDataContext (see page 932)	Sets the data context that provides temporary storage during a call to an invokable interface.
	SubstituteStrings (see page 933)	This is function InvokeRegistry.SubstituteStrings.

23.3.1 InvokeRegistry.GetRemotableDataContext Function

Returns the TDataContext (see page 814) object that provides temporary storage during a call to an invokable interface.

Pascal

```
function GetRemotableDataContext: Pointer;
```

C++

```
void * GetRemotableDataContext();
```

File

InvokeRegistry

Description

GetRemotableDataContext returns a pointer to the data context object that stores all the temporary objects and values needed for implementing a call to an invokable interface. When a client application calls an invokable interface on a Web Service application, the Web Service application automatically creates a data context for the call. Any objects or variables that are needed for marshaling the interface call are automatically added to the data context for the duration of the call. In addition, any remotable objects (TRemotable (see page 868) descendants) that are created while there is a remotable data context available are automatically added to the data context. When the interface call is complete, the data context is freed, and it automatically frees the stored objects and variables.

See Also

SetRemotableDataContext (see page 932)

TRemotable (see page 868)

TDataContext (see page 814)

23.3.2 InvokeRegistry.InvRegistry Function

Provides access to the invocation registry.

Pascal

```
function InvRegistry: TInvokableClassRegistry;
```

C++

```
TInvokableClassRegistry InvRegistry();
```

File

InvokeRegistry

Description

Call InvRegistry to access the invocation registry. The returned object has two methods, RegisterInterface and RegisterInvokableClass, that you can use for registering invokable interfaces (interfaces compiled with RTTI) and the classes that implement them.

Server applications that implement Web Services need to register both the invokable interfaces and the implementation classes. The invoker component can then use the invocation registry to locate invokable interfaces and identify classes that implement them.

Client applications that call on Web Services register the invokable interfaces that they call. The remote interfaced object component can then use the invocation registry to generate a dynamic method table for calling the invokable interfaces.

Note: Both client and server applications must register invokable interfaces using the same namespace. You can either supply this namespace explicitly when you register an interface, or you can use the automatically generated namespace. If you use the automatically generated interface, the interface must be defined in the same unit on both client and server, and both client and server must have the same value set for the global AppNameSpacePrefix (see page 955) variable.

See Also

IInvokable

THTTPSoapPascalInvoker

THTTPRIO

AppNameSpacePrefix (see page 955)

RemTypeRegistry (see page 931)

23.3.3 InvokeRegistry.RemClassRegistry Function

Provides access to the remotable class registry.

Pascal

```
function RemClassRegistry: TRemotableClassRegistry;
```

C++

```
TRemotableClassRegistry RemClassRegistry();
```

File

InvokeRegistry

Description

RemClassRegistry returns the object that implements the remotable class registry. Use its RegisterXSClass method to register a remotable class in an application that publishes or uses Web Services.

Remotable classes are classes that can be transmitted between the client of a Web Services application and the server that implements the service. They can be used to represent non-scalar types for parameters and return values, or they can represent exceptions that occur when trying to execute a request to a Web Service application. All remotable classes are compiled with runtime type information (RTTI) and have a virtual constructor that the remotable class registry recognizes and uses to obtain class instances.

Both the client application and server application must register remotable classes before they can use them. They must register them to use the same namespace. You can either supply this namespace explicitly when you register the class, or you can use the automatically generated namespace. If you use the automatically generated interface, the class must be defined in the same unit on both client and server, and both client and server must have the same value set for the global AppNameSpacePrefix (see page 955) variable.

Note: The remotable class registry is the same thing as the remotable type registry. You can use RemClassRegistry interchangeably with the RemTypeRegistry (see page 931) function.

See Also

TRemotable (see page 868)

ERemotableException (see page 810)

AppNameSpacePrefix (see page 955)

InvRegistry (see page 930)

RemTypeRegistry (see page 931)

23.3.4 InvokeRegistry.RemTypeRegistry Function

Provides access to the remotable type registry.

Pascal

```
function RemTypeRegistry: TRemotableTypeRegistry;
```

C++

```
TRemotableTypeRegistry RemTypeRegistry();
```

File

InvokeRegistry

Description

Call RemTypeRegistry to access to the remotable type registry. Use the RegisterXSClass method of the returned object to register a remotable class in an application that publishes or uses Web Services. Use its RegisterXSInfo method to register a remotable type.

Remotable classes are classes that can be transmitted between the client of a Web Services application and the server that implements the service. They can be used to represent non-scalar types for parameters and return values, or they can represent exceptions that occur when trying to execute a request to a Web Service application. All remotable classes are compiled with

runtime type information (RTTI) and have a virtual constructor that the remotable class registry recognizes and uses to obtain class instances.

Remotable types include dynamic arrays, enumerated types, and boolean types, which can be used in an invokable interface without resorting to a remotable class representation, but which must be registered because their values require special marshaling.

Both the client application and server application must register remotable classes and types before they can use them. They must register them to use the same namespace. You can either supply this namespace explicitly when you register the class or type, or you can use the automatically generated namespace. If you use the automatically generated interface, the class must be defined in the same unit on both client and server, and both client and server must have the same value set for the global AppNameSpacePrefix (see page 955) variable.

Note: The remotable type registry is the same thing as the remotable class registry. You can use RemTypeRegistry interchangeably with the RemClassRegistry (see page 930) function.

See Also

TRemotable (see page 868)

ERemotableException (see page 810)

AppNameSpacePrefix (see page 955)

InvRegistry (see page 930)

23.3.5 InvokeRegistry.SetRemotableDataContext Function

Sets the data context that provides temporary storage during a call to an invokable interface.

Pascal

```
procedure SetRemotableDataContext(Value: Pointer);
```

C++

```
SetRemotableDataContext(void * Value);
```

File

InvokeRegistry

Description

SetRemotableDataContext assigns a data context for a remote call to an invokable interface. Once a data context is assigned, it is available to store the temporary objects and values needed for implementing a call to an invokable interface. If the application creates a TRemotable (see page 868) descendant while there is a remotable data context, that remotable object is automatically added to the data context and freed when the application frees the data context.

Note: The invoker component in a Web Services application automatically calls SetRemotableDataContext to set up a data context for incoming SOAP requests. Thus, there is typically no need to call this method in server applications.

Value is a pointer to a TDataContext (see page 814) descendant that temporarily stores information for remote interface calls.

See Also

GetRemotableDataContext (see page 929)

TRemotable (see page 868)

TDataContext (see page 814)

23.3.6 InvokeRegistry.SubstituteStrings Function

Pascal

```
function SubstituteStrings(const InputString: WideString; const SubString: WideString; const
Replacement: WideString): WideString;
```

C++

```
BSTR SubstituteStrings(const BSTR InputString, const BSTR SubString, const BSTR Replacement);
```

File

InvokeRegistry

Description

This is function InvokeRegistry.SubstituteStrings.

23.4 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	eHeaderMethodType (see page 934)	eHeaderMethodType and THeaderMethodTypeArray (see page 949) indicate the types of messages that should include a given header type.
	ObjectConvertOptions (see page 939)	ObjectConvertOptions and TObjectConvertOptions (see page 950) are flags that customize how to convert a remotable object to or from its SOAP representation.
	SerializationOptions (see page 940)	SerializationOptions and TSerializationOptions (see page 953) are flags that customize how to convert a remotable object to or from its SOAP representation.
	TIntfInvokeOption (see page 942)	TIntfInvokeOption and TIntfInvokeOptions (see page 949) represent flags that customize how to handle calls to an invokable interface.
	TObjMultiOptions (see page 943)	TObjMultiOptions indicates how multiple references to the same remotable class should be translated into a SOAP representation.

Records

	Name	Description
	ExtNameMapItem (see page 935)	This is record InvokeRegistry.ExtNameMapItem.
	IntfExceptionItem (see page 935)	IntfExceptionItem and TExceptionItemArray (see page 948) describe the association between a remotable exception and an interface that can raise it.
	IntfHeaderItem (see page 936)	IntfHeaderItem and THeaderItemArray (see page 948) describe the association between a header and the SOAP messages that include it.
	InvRegClassEntry (see page 937)	This is record InvokeRegistry.InvRegClassEntry.
	InvRegIntfEntry (see page 937)	InvRegIntfEntry stores information about an invokable interface.
	MethParamNameMapItem (see page 939)	This is record InvokeRegistry.MethParamNameMapItem.
	TDynToClear (see page 941)	This is record InvokeRegistry.TDynToClear.
	TRemHeaderEntry (see page 944)	This is record InvokeRegistry.TRemHeaderEntry.
	TRemRegEntry (see page 944)	TRemRegEntry stores information for the remotable type registry.

23.4.1 InvokeRegistry.eHeaderMethodType Enumeration

eHeaderMethodType and THeaderMethodTypeArray (see page 949) indicate the types of messages that should include a given header type.

Pascal

```
eHeaderMethodType = (
  hmtAll,
  hmtRequest,
  hmtResponse
);
```

C++

```
enum eHeaderMethodType {
  hmtAll,
  hmtRequest,
  hmtResponse
};
```

File

InvokeRegistry

Description

THeaderMethodTypeArray (see page 949) is a dynamic array of eHeaderMethodType values. The following table describes the possible eHeaderMethodType values:

Value	Meaning
hmtAll	The header should be included in all messages pertaining to the invokable interface.
hmtRequest	The header should be included in request messages in which the client attempts to invoke the invokable interface.
hmtResponse	The header should be included in response message that the server returns after executing a method on the invokable interface.

23.4.2 InvokeRegistry.ExtNameMapItem Record

Pascal

```
ExtNameMapItem = record
  Name: string;
  ExtName: InvString;
end;
```

C++

```
struct ExtNameMapItem {
  UnicodeString Name;
  InvString ExtName;
};
```

File

InvokeRegistry

Description

This is record InvokeRegistry.ExtNameMapItem.

23.4.3 InvokeRegistry.IntfExceptionItem Record

IntfExceptionItem and TExceptionItemArray (see page 948) describe the association between a remotable exception and an interface that can raise it.

Pascal

```
IntfExceptionItem = record
  ClassType: TClass;
  MethodNames: string;
end;
```

C++

```
struct IntfExceptionItem {
  TClass ClassType;
  UnicodeString MethodNames;
```

```
};
```

File

InvokeRegistry

Description

IntfExceptionItem describes a single remotable exception type. TExceptionItemArray (see page 948) is an array of IntfExceptionItem descriptors. Typically, this array describes the set of exceptions associated with an invokable interface.

ClassType is the class reference for the remotable exception class.

MethodNames is a comma-delimited list of method names. If MethodNames is an empty string, the exception is associated with all the methods on an interface. If MethodNames is nonempty, the exception is associated only with those methods of the interface that the MethodNames field lists.

23.4.4 InvokeRegistry.IntfHeaderItem Record

IntfHeaderItem and THeaderItemArray (see page 948) describe the association between a header and the SOAP messages that include it.

Pascal

```
IntfHeaderItem = record
  Info: PTypeInfo;
  ClassType: TClass;
  Namespace: InvString;
  Name: InvString;
  DefaultMethodType: eHeaderMethodType;
  DefaultRequired: Boolean;
  MethodNames: string;
  MethodTypes: THeaderMethodTypeArray;
  HeaderRequired: TRequiredArray;
end;
```

C++

```
struct IntfHeaderItem {
  PTypeInfo Info;
  TClass ClassType;
  InvString Namespace;
  InvString Name;
  eHeaderMethodType DefaultMethodType;
  Boolean DefaultRequired;
  UnicodeString MethodNames;
  THeaderMethodTypeArray MethodTypes;
  TRequiredArray HeaderRequired;
};
```

File

InvokeRegistry

Description

IntfHeaderItem describes a single header type. THeaderItemArray (see page 948) is an array of IntfHeaderItem descriptors. Typically, this array describes the set of headers associated with an invokable interface.

Info is a pointer to the type description for the interface that the header is associated with.

ClassType is the class reference for the header class (a TSOAPHeader (see page 908) descendant).

NameSpace is the namespace in which the header's name is defined.

Name is the name of the header node.

DefaultMethodType indicates whether the header is included in request messages, response messages, or both. This value is only used if the MethodNames field is an empty string. If MethodNames is nonempty, use the MethodTypes field instead.

DefaultRequired indicates whether the header must appear in messages about the interface that Info describes. If DefaultRequired is false, the header is optional. This value is only used if the MethodNames field is an empty string. If MethodNames is nonempty, use the HeaderRequired field instead.

MethodNames is a comma-delimited list of method names. If empty, the header is associated with all methods on the interface. If MethodNames includes one or more names, the header is only associated with those methods of the interface that are contained in the list.

MethodTypes is a dynamic array of values that indicate whether the header is included in request messages, response messages, or both. Each value is associated with a method listed in the MethodNames field in the same position. If MethodNames is an empty string, MethodTypes is not used.

HeaderRequired is a dynamic array of values that indicate whether the header is required or optional. Each value is associated with a method listed in the MethodNames field in the same position. If MethodNames is an empty string, HeaderRequired is not used.

23.4.5 InvokeRegistry.InvRegClassEntry Record

Pascal

```
InvRegClassEntry = record
  ClassType: TClass;
  Proc: TCreateInstanceProc;
  URI: string;
end;
```

C++

```
struct InvRegClassEntry {
  TClass ClassType;
  TCreateInstanceProc Proc;
  UnicodeString URI;
};
```

File

InvokeRegistry

Description

This is record InvokeRegistry.InvRegClassEntry.

23.4.6 InvokeRegistry.InvRegIntfEntry Record

InvRegIntfEntry stores information about an invokable interface.

Pascal

```
InvRegIntfEntry = record
  Name: string;
  ExtName: InvString;
  UnitName: string;
  GUID: TGUID;
```

```
    Info: PTypeInfo;
    DefImpl: TClass;
    Namespace: InvString;
    WSDLEncoding: InvString;
    Documentation: string;
    SOAPAction: string;
    ReturnParamNames: string;
    InvokeOptions: TIntfInvokeOptions;
    MethNameMap: array of ExtNameMapItem;
    MethParamNameMap: array of MethParamNameMapItem;
    IntfHeaders: array of IntfHeaderItem;
    IntfExceptions: array of IntfExceptionItem;
    UDDIOperator: String;
    UDDIBindingKey: String;
  end;
```

C++

```
  struct InvRegIntfEntry {
    UnicodeString Name;
    InvString ExtName;
    UnicodeString UnitName;
    TGUID GUID;
    PTypeInfo Info;
    TClass DefImpl;
    InvString Namespace;
    InvString WSDLEncoding;
    UnicodeString Documentation;
    UnicodeString SOAPAction;
    UnicodeString ReturnParamNames;
    TIntfInvokeOptions InvokeOptions;
    array of ExtNameMapItem MethNameMap;
    array of MethParamNameMapItem MethParamNameMap;
    array of IntfHeaderItem IntfHeaders;
    array of IntfExceptionItem IntfExceptions;
    UnicodeString UDDIOperator;
    UnicodeString UDDIBindingKey;
  };
```

File

InvokeRegistry

Description

InvRegIntfEntry is used by the invocation registry to store information about a registered invokable interface.

Name is the name of the registered invokable interface.

ExtName is the name of the corresponding port type that appears in the WSDL document describing this Web Service. ExtName can be an empty string if the interface and port type have the same name.

UnitName is the name of the unit in which the interface is defined.

GUID is the globally unique identifier for the interface.

Info points to the runtime type information (RTTI) compiled with the interface.

DefImpl is the default implementation class for the interface.

Namespace is the namespace that is uniquely associated with the interface.

WSDLEncoding is the encoding attribute, if any, of the xml header on a WSDL document that defines the port type to which the invokable interface corresponds.

Documentation is a brief description that documents the interface.

DefaultSOAPAction is the SOAP Action header (or set of SOAP Action headers) to use on messages directed to this interface. If

this is an empty string, the application generates a SOAP Action header that is based on the Namespace field. Assign SOAP Action headers using the RegisterDefaultSOAPAction or RegisterAllSOAPActions method. Note that if you have more than one interface with the same namespace URI, you must register them with different SOAP Action headers so that the server can distinguish them.

ReturnParamNames is a semicolon-delimited list of parameter names that map to the return value of a function.

InvokeOptions are the options for the interface that have been registered using the RegisterInvokeOptions method. These options customize the way the interface is called.

MethNameMap is an array of entries that describe the relationship between methods on the invokable interface and the operation names to which they correspond (if different).

MethParamNameMap is an array of entries that describe the relationship between parameters on a method of the invokable interface and the part names to which they correspond (if different).

IntfHeaders is an array of entries that describe the SOAP headers that should accompany requests directed at the invokable interface.

IntfExceptions is an array of entries that describe the exceptions that the interface can raise.

UDDIOperator is the URL of the UDDI registry, if any, that was used to import the interface definition.

UDDIBindingKey is the unique identifier for the TModel from UDDIOperator that describes the Web Service to which the interface belongs.

23.4.7 InvokeRegistry.MethParamNameMapItem Record

Pascal

```
MethParamNameMapItem = record
  MethName: string;
  ParamNameMap: array of ExtNameMapItem;
end;
```

C++

```
struct MethParamNameMapItem {
  UnicodeString MethName;
  array of ExtNameMapItem ParamNameMap;
};
```

File

InvokeRegistry

Description

This is record InvokeRegistry.MethParamNameMapItem.

23.4.8 InvokeRegistry.ObjectConvertOptions Enumeration

ObjectConvertOptions and TObjectConvertOptions (see page 950) are flags that customize how to convert a remotable object to or from its SOAP representation.

Pascal

```
ObjectConvertOptions = (
```

```
    ocoDontPrefixNode,
    ocoDontSerializeProps,
    ocoDontPutTypeAttr
  );
```

C++

```
  enum ObjectConvertOptions {
    ocoDontPrefixNode,
    ocoDontSerializeProps,
    ocoDontPutTypeAttr
  };
```

File

InvokeRegistry

Description

TObjectConvertOptions (see page 950) is a set of ObjectConvertOptions values. ObjectConvertOptions values are flags that can be supplied when customizing the conversion between an object and its SOAP representation.

Value	Meaning
ocoDontPrefixNode	When set, the SOAP representation does not include a namespace prefix on node names.
ocoDontSerializeProps	When set, the SOAP representation does not include any representation for the published properties of the remotable object.
ocoDontPutTypeAttr	When set, nodes do not include an xsi:type attribute that indicates their type.

23.4.9 InvokeRegistry.SerializationOptions Enumeration

SerializationOptions and TSerializationOptions (see page 953) are flags that customize how to convert a remotable object to or from its SOAP representation.

Pascal

```
  SerializationOptions = (
    xoHolderClass,
    xoAttributeOnLastMember,
    xoInlineArrays,
    xoLiteralParam,
    xoSimpleTypeWrapper,
    xoOption6,
    xoOption7,
    xoOption8,
    xoOption9,
    xoOptionA,
    xoOptionB,
    xoOptionC,
    xoOptionD,
    xoOptionE,
    xoOptionF,
    xoOptionG,
    xoOptionH,
    xoOptionI
  );
```

C++

```
  enum SerializationOptions {
    xoHolderClass,
    xoAttributeOnLastMember,
```

```
    xoInlineArrays,
    xoLiteralParam,
    xoSimpleTypeWrapper,
    xoOption6,
    xoOption7,
    xoOption8,
    xoOption9,
    xoOptionA,
    xoOptionB,
    xoOptionC,
    xoOptionD,
    xoOptionE,
    xoOptionF,
    xoOptionG,
    xoOptionH,
    xoOptionI
  };
```

File

InvokeRegistry

Description

TSerializationOptions (see page 953) is a set of SerializationOptions values. SerializationOptions values are flags that can be registered with an invokable interface to indicate how it should be called. The following table lists the possible values.

Value	Meaning
xoHolderClass	The remotable object corresponds to a "holder" class. That is, the SOAP representation does not include a node for the class itself, just for its members. This is used when a type that would otherwise not require a remotable class uses a feature only available on remotable classes (such as attributes).
xoAttributeOnLastMember	Attributes of the remotable class are stored on the last member of the class that maps to an element node rather than on the node that corresponds to the class itself. This is necessary if a holder class has attributes.
xoInlineArrays	If the remotable class contains class members that are array types, the encoding of those members omits the type name of the array types. This results in more efficient encoding of holder classes that represent array types.
xoLiteralParam	The remotable class represents one of the two top-level nodes (the input or output value) of a document-literal encoding. The class is treated as a holder class if it is used as the direct input or output parameter of a function.
xoSimpleTypeWrapper	The remotable class is a wrapper for a simple type such as a string or integer. The value that the class represents is represented as a single published property of the remotable class.
xoOption5 - xoOptionI	The remaining options are reserved for future use.

23.4.10 InvokeRegistry.TDynToClear Record

Pascal

```
  TDynToClear = record
    P: Pointer;
    Info: PTypeInfo;
  end;
```

C++

```
  struct TDynToClear {
    void * P;
```

```
  PTypeInfo Info;
};
```

File

InvokeRegistry

Description

This is record InvokeRegistry.TDynToClear.

23.4.11 InvokeRegistry.TIntfInvokeOption Enumeration

TIntfInvokeOption and TIntfInvokeOptions (see page 949) represent flags that customize how to handle calls to an invokable interface.

Pascal

```
TIntfInvokeOption = (
  ioDefault,
  ioDocument,
  ioLiteral,
  ioHasDefaultSOAPAction,
  ioHasReturnParamNames,
  ioHasNamespace,
  ioIsAppServerSOAP,
  ioHasUDDIInfo,
  ioHasAllSOAPActions
);
```

C++

```
enum TIntfInvokeOption {
  ioDefault,
  ioDocument,
  ioLiteral,
  ioHasDefaultSOAPAction,
  ioHasReturnParamNames,
  ioHasNamespace,
  ioIsAppServerSOAP,
  ioHasUDDIInfo,
  ioHasAllSOAPActions
};
```

File

InvokeRegistry

Description

TIntfInvokeOptions (see page 949) is a set of TIntfInvokeOption values. TIntfInvokeOption values are flags that can be registered with an invokable interface to indicate how it should be called. The following table lists the possible values.

Value	Meaning
ioDefault	Calls to the interface use the default options unless overridden by other options. (RPC-based SOAP encoding and HTTP as a transport protocol).
ioDocument	The Web Services server uses document-style encoding.
ioLiteral	The Web Services server uses document-literal style and the types for input and output values have not been unwound to create a method call.
ioHasDefaultSOAPAction	A default SOAP action header has been registered for use with the interface.

ioHasReturnParamNames	A set of parameter names have been registered for use with the interface that are reserved for the return value of function calls. If this option is not set, the strings 'Return' and 'Result' are used.
ioHasNamespace	A namespace was explicitly provided when the interface was registered, rather than using an automatically-generated namespace.
ioIsAppServerSOAP	The interface descends from IAppServerSOAP. Such interfaces are used in multi-tier database applications to interact with the application server.
ioHasUDDIInfo	UDDI information has been registered for the interface. This option is set when the Web Service description is imported using the UDDI browser.
ioHasAllSOAPActions	The SOAPAction headers could not be collapsed into a single string. When representing the SOAP action headers, a single string is created where the first character represents a delimiter that marks the boundary between the various SOAP action headers.

23.4.12 InvokeRegistry.TObjMultiOptions Enumeration

TObjMultiOptions indicates how multiple references to the same remotable class should be translated into a SOAP representation.

Pascal

```
TObjMultiOptions = (
  ocDefault,
  ocMultiRef,
  ocNoMultiRef
);
```

C++

```
enum TObjMultiOptions {
  ocDefault,
  ocMultiRef,
  ocNoMultiRef
};
```

File

InvokeRegistry

Description

TObjMultiOptions indicates a class-specific override to the default strategy for converting multiple references to the same object instance into the corresponding SOAP representation. It indicates whether a remotable object should be treated by value (like a structure) or by reference (where object identity is meaningful).

Value	Meaning
ocDefault	Use the default strategy for handling multiple references.
ocMultiRef	Always indicate in the SOAP packet when multiple references refer to the same instance of the specified class.
ocNoMultiRef	Never indicate in the SOAP packet when multiple references refer to the same instance of the specified class. That is, always treat it "by value".

See Also

TRemotableTypeRegistry (see page 874)

23.4.13 InvokeRegistry.TRemHeaderEntry Record

Pascal

```
TRemHeaderEntry = record
  Name: InvString;
  ExtName: InvString;
  URI: InvString;
end;
```

C++

```
struct TRemHeaderEntry {
  InvString Name;
  InvString ExtName;
  InvString URI;
};
```

File

InvokeRegistry

Description

This is record InvokeRegistry.TRemHeaderEntry.

23.4.14 InvokeRegistry.TRemRegEntry Record

TRemRegEntry stores information for the remotable type registry.

Pascal

```
TRemRegEntry = record
  ClassType: TClass;
  Info: PTypeInfo;
  URI: InvString;
  Name: InvString;
  ExtName: InvString;
  IsScalar: Boolean;
  MultiRefOpt: TObjMultiOptions;
  SerializationOpt: TSerializationOptions;
  PropNameMap: array of ExtNameMapItem;
end;
```

C++

```
struct TRemRegEntry {
  TClass ClassType;
  PTypeInfo Info;
  InvString URI;
  InvString Name;
  InvString ExtName;
  Boolean IsScalar;
  TObjMultiOptions MultiRefOpt;
  TSerializationOptions SerializationOpt;
  array of ExtNameMapItem PropNameMap;
};
```

File

InvokeRegistry

Description

TRemRegEntry is the type that the remotable type registry uses to store information about a registered type. The TRemRegEntry type is used to represent both remotable classes and remotable types. It contains the following members:

ClassType is the class reference for the class. It is not used for entries that represent types that are not classes.

Info is the type info pointer for a registered class or type.

URI is the namespace URI that the registry associates with the class or type.

Name is the type name of the class or type, as it appears in encoded method calls or WSDL documents.

ExtName is the same as Name.

IsScalar indicates whether the entry describes a class or type that is marshaled as a scalar type.

MultiRefOpt indicates how to marshal multiple instances of a registered class. It is not used for entries that represent types.

SerializationOpt indicates how to convert between a registered class and its encoded representation.

PropNameMap is an array of entries that describe the relationship between properties on the invokable interface and the external names to which they correspond (if different).

23.5 Types

The following table lists types in this documentation.

Types

Name	Description
ERemotableExceptionClass (see page 946)	This is type InvokeRegistry.ERemotableExceptionClass.
InvString (see page 947)	InvString is the type for strings that are used to represent all or part of an encoded call on an invokable interface.
InvStringDynArray (see page 947)	This is type InvokeRegistry.InvStringDynArray.
PTRemotable (see page 947)	This is type InvokeRegistry.PTRemotable.
TCreateInstanceProc (see page 948)	TCreateInstanceProc is the type of the factory routine that returns an instance of an invokable class.
TExceptionItemArray (see page 948)	This is type InvokeRegistry.TExceptionItemArray.
THeaderItemArray (see page 948)	This is type InvokeRegistry.THeaderItemArray.
THeaderMethodTypeArray (see page 949)	This is type InvokeRegistry.THeaderMethodTypeArray.
TIntfInvokeOptions (see page 949)	This is type InvokeRegistry.TIntfInvokeOptions.
TInvokableClassClass (see page 949)	This is type InvokeRegistry.TInvokableClassClass.
TInvRegIntfEntryArray (see page 950)	This is type InvokeRegistry.TInvRegIntfEntryArray.
TObjectConvertOptions (see page 950)	This is type InvokeRegistry.TObjectConvertOptions.
TPascalRemotableClassRegistry (see page 950)	TPascalRemotableClassRegistry is the type of the remotable class registry.
TPascalRemotableTypeRegistry (see page 951)	TPascalRemotableTypeRegistry is the type of the remotable type registry.
TRemotableClass (see page 951)	This is type InvokeRegistry.TRemotableClass.
TRemotableClassRegistry (see page 951)	This is type InvokeRegistry.TRemotableClassRegistry.
TRemotableXSClass (see page 952)	This is type InvokeRegistry.TRemotableXSClass.
TRemRegEntryArray (see page 952)	This is type InvokeRegistry.TRemRegEntryArray.
TRequiredArray (see page 952)	TRequiredArray is a dynamic array of boolean values.
TSerializationOptions (see page 953)	This is type InvokeRegistry.TSerializationOptions.
TSOAPAttachmentClass (see page 953)	This is type InvokeRegistry.TSOAPAttachmentClass.
TSOAPHeaderClass (see page 953)	This is type InvokeRegistry.TSOAPHeaderClass.

23.5.1 InvokeRegistry.ERemotableExceptionClass Type

Pascal

```
ERemotableExceptionClass = class of ERemotableException;
```

C++

```
ERemotableException ERemotableExceptionClass;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.ERemotableExceptionClass.

23.5.2 InvokeRegistry.InvString Type

InvString is the type for strings that are used to represent all or part of an encoded call on an invokable interface.

Pascal

```
InvString = xmldom.DOMString;
```

C++

```
xmldom.DOMString InvString;
```

File

InvokeRegistry

Description

InvString is the type for encoded method calls in Web Service applications. It is used for the results of encoding an interface call or its results so that it can be transported between client and server applications.

23.5.3 InvokeRegistry.InvStringDynArray Type

Pascal

```
InvStringDynArray = array of InvString;
```

C++

```
array of InvString InvStringDynArray;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.InvStringDynArray.

23.5.4 InvokeRegistry.PTRemotable Type

Pascal

```
PTRemotable = ^TRemotable;
```

C++

```
^TRemotable PTRemotable;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.PTRemotable.

23.5.5 InvokeRegistry.TCreateInstanceProc Type

TCreateInstanceProc is the type of the factory routine that returns an instance of an invokable class.

Pascal

```
TCreateInstanceProc = procedure (out obj: TObject);
```

C++

```
(out obj: TObject) ( TCreateInstanceProc)();
```

File

InvokeRegistry

Description

TCreateInstanceProc is the type for procedures that return an instance of an invokable class on behalf of the invocation registry. When the class that implements an invokable interface is not a descendant of TInvokableClass (see page 833), or when you want to control the instantiation of the implementation class (for example to use a single shared instance for all incoming requests), supply a procedure of this type when registering the implementation class.

obj returns an instance of the invokable class, which can handle the current incoming request.

23.5.6 InvokeRegistry.TExceptionItemArray Type

Pascal

```
TExceptionItemArray = array of IntfExceptionItem;
```

C++

```
array of IntfExceptionItem TExceptionItemArray;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TExceptionItemArray.

23.5.7 InvokeRegistry.THeaderItemArray Type

Pascal

```
THeaderItemArray = array of IntfHeaderItem;
```

C++

```
array of IntfHeaderItem THeaderItemArray;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.THeaderItemArray.

23.5.8 InvokeRegistry.THeaderMethodTypeArray Type

Pascal

```
THeaderMethodTypeArray = array of eHeaderMethodType;
```

C++

```
array of eHeaderMethodType THeaderMethodTypeArray;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.THeaderMethodTypeArray.

23.5.9 InvokeRegistry.TIntfInvokeOptions Type

Pascal

```
TIntfInvokeOptions = set of TIntfInvokeOption;
```

C++

```
TIntfInvokeOption TIntfInvokeOptions;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TIntfInvokeOptions.

23.5.10 InvokeRegistry.TInvokableClassClass Type

Pascal

```
TInvokableClassClass = class of TInvokableClass;
```

C++

```
TInvokableClass TInvokableClassClass;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TInvokableClassClass.

23.5.11 InvokeRegistry.TInvRegIntfEntryArray Type

Pascal

```
TInvRegIntfEntryArray = array of InvRegIntfEntry;
```

C++

```
array of InvRegIntfEntry TInvRegIntfEntryArray;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TInvRegIntfEntryArray.

23.5.12 InvokeRegistry.TObjectConvertOptions Type

Pascal

```
TObjectConvertOptions = set of ObjectConvertOptions;
```

C++

```
ObjectConvertOptions TObjectConvertOptions;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TObjectConvertOptions.

23.5.13 InvokeRegistry.TPascalRemotableClassRegistry Type

TPascalRemotableClassRegistry is the type of the remotable class registry.

Pascal

```
TPascalRemotableClassRegistry = TRemotableTypeRegistry;
```

C++

```
TRemotableTypeRegistry TPascalRemotableClassRegistry;
```

File

InvokeRegistry

Description

TPascalRemotableClassRegistry is a synonym for TRemotableTypeRegistry (see page 874). It is defined for compatibility with Delphi 6.

See Also

RemClassRegistry (see page 930)

TRemotable (see page 868)

23.5.14 InvokeRegistry.TPascalRemotableTypeRegistry Type

TPascalRemotableTypeRegistry is the type of the remotable type registry.

Pascal

```
TPascalRemotableTypeRegistry = TRemotableTypeRegistry;
```

C++

```
TRemotableTypeRegistry TPascalRemotableTypeRegistry;
```

File

InvokeRegistry

Description

TPascalRemotableTypeRegistry is a synonym for TRemotableTypeRegistry (see page 874). It is defined for compatibility with Delphi 6.

See Also

RemTypeRegistry (see page 931)

TRemotable (see page 868)

23.5.15 InvokeRegistry.TRemotableClass Type

Pascal

```
TRemotableClass = class of TRemotable;
```

C++

```
TRemotable TRemotableClass;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TRemotableClass.

23.5.16 InvokeRegistry.TRemotableClassRegistry Type

Pascal

```
TRemotableClassRegistry = TRemotableTypeRegistry;
```

C++

```
TRemotableTypeRegistry TRemotableClassRegistry;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TRemotableClassRegistry.

23.5.17 InvokeRegistry.TRemotableXSClass Type

Pascal

```
TRemotableXSClass = class of TRemotableXS;
```

C++

```
TRemotableXS TRemotableXSClass;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TRemotableXSClass.

23.5.18 InvokeRegistry.TRemRegEntryArray Type

Pascal

```
TRemRegEntryArray = array of TRemRegEntry;
```

C++

```
array of TRemRegEntry TRemRegEntryArray;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TRemRegEntryArray.

23.5.19 InvokeRegistry.TRequiredArray Type

TRequiredArray is a dynamic array of boolean values.

Pascal

```
TRequiredArray = array of Boolean;
```

C++

```
array of Boolean TRequiredArray;
```

File

InvokeRegistry

Description

TRequiredArray is a dynamic array of boolean values. Each value indicates whether a corresponding item is required (true) or optional (false).

23.5.20 InvokeRegistry.TSerializationOptions Type

Pascal

```
TSerializationOptions = set of SerializationOptions;
```

C++

```
SerializationOptions TSerializationOptions;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TSerializationOptions.

23.5.21 InvokeRegistry.TSOAPAttachmentClass Type

Pascal

```
TSOAPAttachmentClass = class of TSOAPAttachment;
```

C++

```
TSOAPAttachment TSOAPAttachmentClass;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TSOAPAttachmentClass.

23.5.22 InvokeRegistry.TSOAPHeaderClass Type

Pascal

```
TSOAPHeaderClass = class of TSOAPHeader;
```

C++

```
TSOAPHeader TSOAPHeaderClass;
```

File

InvokeRegistry

Description

This is type InvokeRegistry.TSOAPHeaderClass.

23.6 Variables

The following table lists variables in this documentation.

Variables

Name	Description
AppNameSpacePrefix (see page 955)	Supplies a unique prefix to use for automatically generated namespaces given to invokable interfaces and remotable types.

23.6.1 InvokeRegistry.AppNameSpacePrefix Variable

Supplies a unique prefix to use for automatically generated namespaces given to invokable interfaces and remotable types.

Pascal

```
AppNameSpacePrefix: string;
```

C++

```
UnicodeString AppNameSpacePrefix;
```

File

InvokeRegistry

Description

Set AppNameSpacePrefix to a unique string that the invocation registry and remotable type registry use when generating namespaces. These generated namespaces must be unique strings that identify invokable interfaces and remotable classes in a Web Service application. They are generated by the RegisterInterface method of the invocation registry and the RegisterXSClass and RegisterXSInfo methods of the remotable class registry if the application does not explicitly supply a namespace.

The invocation registry and remotable type registry generate namespaces using the value of AppNameSpacePrefix, the name of the unit in which the interface or class is defined, and the name of the class, type, or interface. If your unit and interface or class names are at all generic, it is a good idea to supply a unique value to AppNameSpacePrefix to guard against the possibility of another interface, type, or class with the same name that is registered in another unit with the same name.

Both client and server applications must register invokable interfaces and remotable types and classes using the same namespace. If you rely on automatically-generated namespaces, be sure to use the same value of AppNameSpacePrefix in both the client and the server applications.

See Also

InvRegistry (see page 930)

TInvokableClassRegistry (see page 840)

TRemotableTypeRegistry (see page 874)

RemTypeRegistry (see page 931)

23.7 Constants

The following table lists constants in this documentation.

Constants

Name	Description
AS_ATTRIBUTE (see page 956)	Identifies a property of a remotable class as one that corresponds to an attribute in the SOAP encoding.
XMLBase64Types (see page 956)	This is constant InvokeRegistry.XMLBase64Types.
XMLSchemaInstNamepspaces (see page 957)	This is constant InvokeRegistry.XMLSchemaInstNamepspaces.
XMLSchemaNamepspaces (see page 957)	This is constant InvokeRegistry.XMLSchemaNamepspaces.

23.7.1 InvokeRegistry.AS_ATTRIBUTE Constant

Identifies a property of a remotable class as one that corresponds to an attribute in the SOAP encoding.

Pascal

```
AS_ATTRIBUTE = False;
```

C++

```
AS_ATTRIBUTE = False;
```

File

InvokeRegistry

Description

Assign AS_ATTRIBUTE as the value of the stored directive on a property declaration in a TRemotable (see page 868) descendant to indicate that the property is encoded as an attribute node rather than as an element node. If a property declaration does not include a stored directive with a value of AS_ATTRIBUTE, it is encoded as an element node instead.

Note: Note:AS_ATTRIBUTE must be used directly. If you set the stored directive to a function that returns AS_ATTRIBUTE (false), the property is still encoded as an element node.

See Also

TRemotable (see page 868)

TRemotableXS (see page 892)

23.7.2 InvokeRegistry.XMLBase64Types Constant

Pascal

```
XMLBase64Types: array[0..1] of InvString = ('base64Binary', 'bin.base64');
```

C++

```
array[0..1] of InvString XMLBase64Types = ('base64Binary', 'bin.base64');
```

File

InvokeRegistry

Description

This is constant InvokeRegistry.XMLBase64Types.

23.7.3 InvokeRegistry.XMLSchemaInstNamepspaces Constant

Pascal

```
XMLSchemaInstNamepspaces: array[0..2] of InvString = (SXMLSchemaInstURI_1999,
SXMLSchemaInstURI_2000_10, SXMLSchemaInstURI);
```

C++

```
array[0..2] of InvString XMLSchemaInstNamepspaces = (SXMLSchemaInstURI_1999,
SXMLSchemaInstURI_2000_10, SXMLSchemaInstURI);
```

File

InvokeRegistry

Description

This is constant InvokeRegistry.XMLSchemaInstNamepspaces.

23.7.4 InvokeRegistry.XMLSchemaNamepspaces Constant

Pascal

```
XMLSchemaNamepspaces: array[0..2] of InvString = (SXMLSchemaURI_1999, SXMLSchemaURI_2000_10,
SXMLSchemaURI_2001);
```

C++

```
array[0..2] of InvString XMLSchemaNamepspaces = (SXMLSchemaURI_1999, SXMLSchemaURI_2000_10,
SXMLSchemaURI_2001);
```

File

InvokeRegistry

Description

This is constant InvokeRegistry.XMLSchemaNamepspaces.

24 ISAPIApp Namespace

This is namespace ISAPIApp.

Topics

Name	Description
Classes (see page 960)	The following table lists classes in this documentation.
Functions (see page 965)	The following table lists functions in this documentation.
Variables (see page 967)	The following table lists variables in this documentation.

Classes

	Name	Description
	TISAPIApplication (see page 960)	TISAPIApplication encapsulates an ISAPI or NSAPI server application.
	TISAPIFactory (see page 963)	

Functions

	Name	Description
	GetExtensionVersion (see page 965)	Returns the name and version number of an ISAPI or NSAPI application.
	HttpExtensionProc (see page 965)	Receives an HTTP request message for an ISAPI or NSAPI application.
	TerminateExtension (see page 966)	Indicates whether an ISAPI or NSAPI application is ready to be shut down.

Variables

Name	Description
DispatchThread (see page 967)	This is variable ISAPIApp.DispatchThread.

24.1 Classes

The following table lists classes in this documentation.

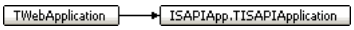
Classes

	Name	Description
	TISAPIApplication (see page 960)	TISAPIApplication encapsulates an ISAPI or NSAPI server application.
	TISAPIFactory (see page 963)	

24.1.1 TISAPIApplication Class

TISAPIApplication encapsulates an ISAPI or NSAPI server application.

Class Hierarchy



Pascal

```
TISAPIApplication = class(TWebApplication);
```

C++

```
class TISAPIApplication : public TWebApplication;
```

File

ISAPIApp

Description

TISAPIApplication provides the fundamental behavior of an ISAPI or NSAPI server application. The HTTP server calls the methods of TISAPIApplication when it receives an HTTP request message. TISAPIApplication creates a thread to manage the request message, and a creates a new instance of the Web module for that thread, or reuses an old instance that was deactivated after a previous response was sent.

TISAPIApplication creates TISAPIRequest (see page 970) and TISAPIResponse (see page 977) objects to represent the request message and any response that should be sent in return, passes these to the dispatcher so that the response can be filled in, and sends the response if it was not already sent by a TWebActionItem.

Each ISAPI server project automatically declares an Application variable of type TISAPIApplication as the instance of the application. This declaration overrides the Application variable declared in the Forms unit if the ISAPIApp (see page 959) unit is declared last in the uses list of the project file. Every ISAPI server application must include the ISAPIApp (see page 959) unit.

To work with an Apache server, use the TApacheApplication object. To work with a CGI server, use the TCGIApplication object instead.

See Also

TCGIApplication

TISAPIRequest (see page 970)

TISAPIResponse (see page 977)

TWebActionItem

TWebDispatcher

24.1.1.1 TISAPIApplication Members

The following tables list the members exposed by TISAPIApplication.

Public Methods

	Name	Description
	Create (see page 961)	Creates an instance of TWebApplication.
	Destroy (see page 961)	Destroys an instance of TWebApplication.
	GetExtensionVersion (see page 962)	Returns the application name and version number.
	HttpExtensionProc (see page 962)	Responds to an HTTP request message.
	TerminateExtension (see page 963)	Returns true.

24.1.1.2 TISAPIApplication.Create Constructor

Creates an instance of TWebApplication.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TISAPIApplication(TComponent * AOwner);
```

Description

Do not call Create. The single Web application object named Application is instantiated automatically when the CGIApp unit, the ISAPIApp unit, or the ApacheApp unit is included in the uses list of the project.

Create allocates the helper objects used by TWebApplication.

See Also

Create

24.1.1.3 TISAPIApplication.Destroy Destructor

Destroys an instance of TWebApplication.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TISAPIApplication();
```

Description

Never call the Destroy or Free method. The application is freed automatically when the application stops running.

Destroy frees the helper objects that were created at startup, as well as all Web modules created by the application.

See Also

System::TObject::Free

Run

24.1.1.4 TISAPIApplication Methods

The methods of the TISAPIApplication class are listed here.

Public Methods

	Name	Description
	GetExtensionVersion (see page 962)	Returns the application name and version number.
	HttpExtensionProc (see page 962)	Responds to an HTTP request message.
	TerminateExtension (see page 963)	Returns true.

24.1.1.4.1 TISAPIApplication.GetExtensionVersion Method

Returns the application name and version number.

Pascal

```
function GetExtensionVersion(var Ver: THSE_VERSION_INFO): BOOL;
```

C++

```
__fastcall BOOL GetExtensionVersion(THSE_VERSION_INFO Ver);
```

Description

The HTTP server calls GetExtensionVersion to obtain the application name and version number. GetExtensionVersion returns the version number for the HTTP functions in the runtime library, and the Title property of the ISAPI application object.

See Also

Title

24.1.1.4.2 TISAPIApplication.HttpExtensionProc Method

Responds to an HTTP request message.

Pascal

```
function HttpExtensionProc(var ECB: TEXTENSION_CONTROL_BLOCK): DWORD;
```

C++

```
__fastcall DWORD HttpExtensionProc(TEXTENSION_CONTROL_BLOCK ECB);
```

Description

The HTTP server calls HttpExtensionProc when it receives an HTTP request message. HttpExtensionProc creates the TISAPIRequest (see page 970) and TISAPIResponse (see page 977) objects for the HTTP request message, and calls the HandleRequest message so that the response can be filled out and sent. Finally, HttpExtensionProc frees the request and response objects.

If the response is filled out and sent, HttpExtensionProc returns HSE_STATUS_SUCCESS. Otherwise it returns HSE_STATUS_ERROR.

See Also

TISAPIRequest (see page 970)

TISAPIResponse (see page 977)

HandleRequest

TWebDispatcher

24.1.1.4.3 TISAPIApplication.TerminateExtension Method

Returns true.

Pascal

```
function TerminateExtension(dwFlags: DWORD): BOOL;
```

C++

```
__fastcall BOOL TerminateExtension(DWORD dwFlags);
```

Description

The HTTP server calls TerminateExtension when it wants to shut down the ISAPI application. TerminateExtension always returns true, allowing the server to shut down.

24.1.2 TISAPIFactory Class

Class Hierarchy

Pascal

```
TISAPIFactory = class(TObject);
```

C++

```
class TISAPIFactory : public TObject;
```

File

ISAPIApp

See Also

TCGIApplication

24.1.2.1 TISAPIFactory Members

The following tables list the members exposed by TISAPIFactory.

Protected Methods

	Name	Description
	NewRequest (see page 964)	This is NewRequest, a member of class TISAPIFactory.
	NewResponse (see page 964)	This is NewResponse, a member of class TISAPIFactory.

Public Methods

	Name	Description
	Create (see page 963)	Creates an instance of Create. {Placeholder—Update for BCB}

24.1.2.2 TISAPIFactory.Create Constructor

Creates an instance of Create. {Placeholder—Update for BCB}

Pascal

```
constructor Create;
```

C++

```
__fastcall TISAPIFactory();
```

Description

Do not create instances of Create. Instead, use the new keyword, passing any arguments for the Create's constructor.

See Also

System::TObject::Destroy

new

24.1.2.3 TISAPIFactory Methods

The methods of the TISAPIFactory class are listed here.

Protected Methods

	Name	Description
	NewRequest (see page 964)	This is NewRequest, a member of class TISAPIFactory.
	NewResponse (see page 964)	This is NewResponse, a member of class TISAPIFactory.

24.1.2.3.1 TISAPIFactory.NewRequest Method

Pascal

```
function NewRequest(var AECB: TEXTENSION_CONTROL_BLOCK): TISAPIRequest; virtual;
```

C++

```
virtual __fastcall TISAPIRequest NewRequest(TEXTENSION_CONTROL_BLOCK AECB);
```

Description

This is NewRequest, a member of class TISAPIFactory.

24.1.2.3.2 TISAPIFactory.NewResponse Method

Pascal

```
function NewResponse(ISAPIRequest: TISAPIRequest): TISAPIResponse; virtual;
```

C++

```
virtual __fastcall TISAPIResponse NewResponse(TISAPIRequest ISAPIRequest);
```

Description

This is NewResponse, a member of class TISAPIFactory.

24.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	GetExtensionVersion (see page 965)	Returns the name and version number of an ISAPI or NSAPI application.
	HttpExtensionProc (see page 965)	Receives an HTTP request message for an ISAPI or NSAPI application.
	TerminateExtension (see page 966)	Indicates whether an ISAPI or NSAPI application is ready to be shut down.

24.2.1 ISAPIApp.GetExtensionVersion Function

Returns the name and version number of an ISAPI or NSAPI application.

Pascal

```
function GetExtensionVersion(var Ver: THSE_VERSION_INFO): BOOL; stdcall;
```

C++

```
__stdcall BOOL GetExtensionVersion(THSE_VERSION_INFO Ver);
```

File

ISAPIApp

Description

Do not call GetExtensionVersion. ISAPI and NSAPI DLLs export GetExtension to provide an entry point that is called by ISAPI or NSAPI Web servers. This allows the Web server to obtain version information about the web server application.

See Also

HttpExtensionProc (see page 965)

TerminateExtension (see page 966)

TISAPIApplication (see page 960)

24.2.2 ISAPIApp.HttpExtensionProc Function

Receives an HTTP request message for an ISAPI or NSAPI application.

Pascal

```
function HttpExtensionProc(var ECB: TEXTENSION_CONTROL_BLOCK): DWORD; stdcall;
```

C++

```
__stdcall DWORD HttpExtensionProc(TEXTENSION_CONTROL_BLOCK ECB);
```

File

ISAPIApp

Description

Do not call HttpExtensionProc. ISAPI and NSAPI DLLs export HttpExtensionProc to provide an entry point that is called by ISAPI or NSAPI Web servers when they receive HTTP request messages.

See Also

GetExtensionVersion (see page 965)

TerminateExtension (see page 966)

24.2.3 ISAPIApp.TerminateExtension Function

Indicates whether an ISAPI or NSAPI application is ready to be shut down.

Pascal

```
function TerminateExtension(dwFlags: DWORD): BOOL; stdcall;
```

C++

```
__stdcall BOOL TerminateExtension(DWORD dwFlags);
```

File

ISAPIApp

Description

Do not call TerminateExtension. TerminateExtension is exported by ISAPI and NSAPI DLLs. It provides an entry point that is called by ISAPI or NSAPI Web servers, when they are ready to shut down the ISAPI or NSAPI application, to check whether it would cause problems to do so.

See Also

HttpExtensionProc (see page 965)

24.3 Variables

The following table lists variables in this documentation.

Variables

Name	Description
DispatchThread (see page 967)	This is variable ISAPIApp.DispatchThread.

24.3.1 ISAPIApp.DispatchThread Variable

Pascal

```
DispatchThread: function (AECB: PEXTENSION_CONTROL_BLOCK): Boolean of object;
```

C++

```
function (AECB: PEXTENSION_CONTROL_BLOCK): Boolean of object DispatchThread;
```

File

ISAPIApp

Description

This is variable ISAPIApp.DispatchThread.

25 IsapiHTTP Namespace

This is namespace IsapiHTTP.

Topics

Name	Description
Classes (see page 970)	The following table lists classes in this documentation.

Classes

	Name	Description
	TISAPIRequest (see page 970)	TISAPIRequest represents client request information that is passed to a server application DLL.
	TISAPIResponse (see page 977)	TISAPIResponse represents HTTP messages sent by an ISAPI application in response to an HTTP request message.

25.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TISAPIRequest (see page 970)	TISAPIRequest represents client request information that is passed to a server application DLL.
	TISAPIResponse (see page 977)	TISAPIResponse represents HTTP messages sent by an ISAPI application in response to an HTTP request message.

25.1.1 TISAPIRequest Class

TISAPIRequest represents client request information that is passed to a server application DLL.

Class Hierarchy

Pascal

```
TISAPIRequest = class(TWebRequest);
```

C++

```
class TISAPIRequest : public TWebRequest;
```

File

IsapiHTTP

Description

Use TISAPIRequest to interpret a client request to a server application DLL.

ISAPI application objects create TISAPIRequest objects to capture client requests. The properties of the TISAPIRequest objects are obtained from the Web server. The ISAPI application uses TISAPIRequest objects to create corresponding TISAPIResponse (see page 977) objects. The TWebDispatcher for the application then invokes the TWebActionItem that is associated with the path portion of the URL of the TISAPIRequest object.

See Also

TISAPIApplication (see page 960)

TISAPIResponse (see page 977)

TWebActionItem

TWebDispatcher

Accessing Client Request Information

25.1.1.1 TISAPIRequest Members

The following tables list the members exposed by TISAPIRequest.

Protected Methods

	Name	Description
	GetDateVariable (see page 972)	Extracts a date value from the extension control block.
	GetIntegerVariable (see page 973)	Extracts an integer value from the extension control block.
	GetStringVariable (see page 973)	Extracts a string value from the extension control block.

Public Methods

	Name	Description
	Create (see page 971)	Creates an instance of TISAPIRequest (see page 970) to represent the request message information contained in the AECB parameter.
	GetFieldByName (see page 972)	Returns the value of the HTTP header variable specified by the Name parameter.
	ReadClient (see page 974)	Reads Count bytes from the HTTP request into Buffer.
	ReadString (see page 974)	Reads Count bytes from the HTTP request into a string.
	TranslateURI (see page 975)	Translates a Uniform Resource Identifier (URI) into the internal representation of a location on the server.
	WriteClient (see page 975)	Writes Count bytes from Buffer back to the HTTP client.
	WriteHeaders (see page 975)	Writes the header contents back to the HTTP client.
	WriteString (see page 976)	Writes the string AString back to the HTTP client.

Public Properties

	Name	Description
	ECB (see page 976)	Indicates the extension control block structure used by the server application DLL to communicate with the Web server.

25.1.1.2 TISAPIRequest.Create Constructor

Creates an instance of TISAPIRequest (see page 970) to represent the request message information contained in the AECB parameter.

Pascal

```
constructor Create(AECB: PEXTENSION_CONTROL_BLOCK);
```

C++

```
__fastcall TISAPIRequest(PEXTENSION_CONTROL_BLOCK AECB);
```

Description

Applications don't need to call Create to instantiate an ISAPI request object. TISAPIApplication creates a TISAPIRequest (see page 970) object whenever it receives a client request.

See Also

ECB (see page 976)

TISAPIApplication (see page 960)

25.1.1.3 TISAPIRequest Methods

The methods of the TISAPIRequest class are listed here.

Protected Methods

	Name	Description
	GetDateVariable (see page 972)	Extracts a date value from the extension control block.
	GetIntegerVariable (see page 973)	Extracts an integer value from the extension control block.
	GetStringVariable (see page 973)	Extracts a string value from the extension control block.

Public Methods

	Name	Description
	GetFieldByName (see page 972)	Returns the value of the HTTP header variable specified by the Name parameter.
	ReadClient (see page 974)	Reads Count bytes from the HTTP request into Buffer.
	ReadString (see page 974)	Reads Count bytes from the HTTP request into a string.
	TranslateURI (see page 975)	Translates a Uniform Resource Identifier (URI) into the internal representation of a location on the server.
	WriteClient (see page 975)	Writes Count bytes from Buffer back to the HTTP client.
	WriteHeaders (see page 975)	Writes the header contents back to the HTTP client.
	WriteString (see page 976)	Writes the string AString back to the HTTP client.

25.1.1.3.1 TISAPIRequest.GetDateVariable Method

Extracts a date value from the extension control block.

Pascal

```
function GetDateVariable(Index: Integer): TDateTime; override;
```

C++

```
virtual __fastcall TDateTime GetDateVariable(int Index);
```

Description

GetDateVariable reads date-time values from the extension control block specified by the ECB (see page 976) property. The ISAPI request object uses GetDateVariable to access the properties that represent dates from the HTTP request headers.

See Also

Date

ECB (see page 976)

Expires

IfModifiedSince

25.1.1.3.2 TISAPIRequest.GetFieldByName Method

Returns the value of the HTTP header variable specified by the Name parameter.

Pascal

```
function GetFieldByName(const Name: AnsiString): AnsiString; override;
```

C++

```
virtual __fastcall AnsiString GetFieldByName(const AnsiString Name);
```

Description

Use GetFieldByName to read the value of an HTTP header variable that is not surfaced as one of the properties of TISAPIRequest (see page 970). For information on possible HTTP request header variables, see the current Internet standards documents.

See Also

ECB (see page 976)

GetStringVariable (see page 973)

25.1.1.3.3 TISAPIRequest.GetIntegerVariable Method

Extracts an integer value from the extension control block.

Pascal

```
function GetIntegerVariable(Index: Integer): Integer; override;
```

C++

```
virtual __fastcall int GetIntegerVariable(int Index);
```

Description

GetIntegerVariable reads integer values from the extension control block specified by the ECB (see page 976) property. The ISAPI request object uses GetIntegerVariable to access the properties that represent integers from the HTTP request headers.

See Also

ContentLength

ECB (see page 976)

ServerPort

25.1.1.3.4 TISAPIRequest.GetStringVariable Method

Extracts a string value from the extension control block.

Pascal

```
function GetStringVariable(Index: Integer): AnsiString; override;
```

C++

```
virtual __fastcall AnsiString GetStringVariable(int Index);
```

Description

GetStringVariable reads string values from the extension control block specified by the ECB (see page 976) property. The ISAPI request object uses GetStringVariable to access the properties that represent strings from the HTTP request headers.

See Also

Accept

CacheControl

Connection

Content

ContentEncoding

ContentType

ContentVersion

Cookie

DerivedFrom

ECB (see page 976)

From

Host

TISAPIRequest (see page 970)

PathInfo

PathTranslated

ProtocolVersion

Query

Referer

RemoteAddr

RemoteHost

ScriptName

Title

URL

UserAgent

25.1.1.3.5 TISAPIRequest.ReadClient Method

Reads Count bytes from the HTTP request into Buffer.

Pascal

```
function ReadClient(var Buffer; Count: Integer): Integer; override;
```

C++

```
virtual __fastcall int ReadClient( Buffer, int Count);
```

Description

Use ReadClient to load a chunk of information from the contents of an HTTP request into a buffer. ReadClient allows the server application to get the next chunk of information when the content of the request message is too large for the Content property. ReadClient returns the number of bytes actually read. If no more content is available, ReadClient returns -1.

See Also

ReadString (see page 974)

WriteClient (see page 975)

25.1.1.3.6 TISAPIRequest.ReadString Method

Reads Count bytes from the HTTP request into a string.

Pascal

```
function ReadString(Count: Integer): AnsiString; override;
```

C++

```
virtual __fastcall AnsiString ReadString(int Count);
```

Description

Use ReadString to load the next chunk of content information form the HTTP request into a string. ReadString allows the server application to get the next chunk of information when the content of the request message is too large for the Content property.

See Also

GetStringVariable (see page 973)

ReadClient (see page 974)

WriteString (see page 976)

25.1.1.3.7 TISAPIRequest.TranslateURI Method

Translates a Uniform Resource Identifier (URI) into the internal representation of a location on the server.

Pascal

```
function TranslateURI(const URI: string): string; override;
```

C++

```
virtual __fastcall UnicodeString TranslateURI(const UnicodeString URI);
```

Description

Use TranslateURI is used to translate the URI specified in a client request into a format that can be used by the server application.

See Also

PathTranslated

URL

25.1.1.3.8 TISAPIRequest.WriteClient Method

Writes Count bytes from Buffer back to the HTTP client.

Pascal

```
function WriteClient(var Buffer; Count: Integer): Integer; override;
```

C++

```
virtual __fastcall int WriteClient( Buffer, int Count);
```

Description

Use WriteClient to write the contents of a response back to the Web client.

See Also

ReadClient (see page 974)

TISAPIResponse (see page 977)

WriteString (see page 976)

WriteHeaders (see page 975)

25.1.1.3.9 TISAPIRequest.WriteHeaders Method

Writes the header contents back to the HTTP client.

Pascal

```
function WriteHeaders(StatusCode: Integer; const StatusString: AnsiString; const Headers:
AnsiString): Boolean; override;
```

C++

```
virtual __fastcall Boolean WriteHeaders(int StatusCode, const AnsiString StatusString, const AnsiString Headers);
```

Description

WriteHeaders writes the HTTP headers of a response message before a call to the WriteClient (see page 975) method. These headers are used on the HTTP message that the WriteClient (see page 975) method sends in response to this request.

StatusCode is the status code of the response. This informs the client of the status of the request. For a list of status code values, see the StatusCode property of TWebResponse.

StatusString is a text string that describes the status that StatusCode indicates.

Headers is a formatted string that includes any other HTTP headers to add. It represents each header as a string of the form 'Headername: Value', where name/value pairs are separated by line breaks.

WriteHeaders returns true if it succeeds in writing the specified headers.

See Also

WriteClient (see page 975)

StatusCode

25.1.1.3.10 TISAPIRequest.WriteString Method

Writes the string AString back to the HTTP client.

Pascal

```
function WriteString(const AString: AnsiString): Boolean; override;
```

C++

```
virtual __fastcall Boolean WriteString(const AnsiString AString);
```

Description

Use WriteString to write the contents of a response back to the Web client.

See Also

ReadString (see page 974)

TISAPIResponse (see page 977)

WriteClient (see page 975)

25.1.1.4 TISAPIRequest Properties

The properties of the TISAPIRequest class are listed here.

Public Properties

	Name	Description
R	ECB (see page 976)	Indicates the extension control block structure used by the server application DLL to communicate with the Web server.

25.1.1.4.1 TISAPIRequest.ECB Property

Indicates the extension control block structure used by the server application DLL to communicate with the Web server.

Pascal

```
property ECB: PEXTENSION_CONTROL_BLOCK;
```

C++

```
__property PEXTENSION_CONTROL_BLOCK ECB;
```

Description

Read ECB to gain direct access to the extension control block that the server application DLL uses to represent the client request message. Most applications will not need to use the ECB property because the header fields values can be read using the GetFieldByName (see page 972) method.

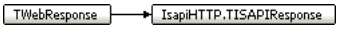
See Also

GetFieldByName (see page 972)

25.1.2 TISAPIResponse Class

TISAPIResponse represents HTTP messages sent by an ISAPI application in response to an HTTP request message.

Class Hierarchy



Pascal

```
TISAPIResponse = class(TWebResponse);
```

C++

```
class TISAPIResponse : public TWebResponse;
```

File

IsapiHTTP

Description

Use TISAPIResponse to specify the response to an HTTP request message.

An ISAPI application creates a TISAPIResponse object based on the TISAPIRequest (see page 970) object for an incoming HTTP request message. TWebDispatcher then passes the TISAPIResponse object to the TWebActionItem associated with the TISAPIRequest (see page 970) object. The TWebActionItem sets the properties of the TISAPIResponse object.

See Also

TISAPIApplication (see page 960)

TISAPIRequest (see page 970)

TWebActionItem

TWebDispatcher

Creating HTTP Response Messages

25.1.2.1 TISAPIResponse Members

The following tables list the members exposed by TISAPIResponse.

Protected Methods

	Name	Description
	GetContent (see page 979)	Reads the value of the Content property.
	GetDateVariable (see page 979)	Extracts a date from the internal table of HTTP header variables.
	GetIntegerVariable (see page 980)	Extracts an integer value from the internal table of HTTP header variables.
	GetLogMessage (see page 980)	Reads the value of the LogMessage property.
	GetStatusCode (see page 980)	Reads the value of the StatusCode property.
	GetStringVariable (see page 981)	Extracts a string value from the internal table of HTTP header variables.
	InitResponse (see page 981)	This is InitResponse, a member of class TISAPIResponse.
	SetContent (see page 983)	Sets the Content property.
	SetDateVariable (see page 983)	Sets a date-time valued HTTP header property.
	SetIntegerVariable (see page 984)	Sets an integer valued HTTP header property.
	SetLogMessage (see page 984)	Sets the value of the LogMessage property.
	SetStatusCode (see page 984)	Sets the StatusCode property.
	SetStringVariable (see page 985)	Sets a string valued HTTP header property.

Public Methods

	Name	Description
	Create (see page 978)	Creates an instance of TISAPIResponse (see page 977) to represent the response to the HTTPRequest parameter.
	SendRedirect (see page 982)	Sends the response with a StatusCode of 301 (Moved Permanently) and with Location set to the URI parameter.
	SendResponse (see page 982)	Sends the ISAPI response message to the web client.
	SendStream (see page 982)	Sends the ISAPI response message with the contents set to the contents of AStream.
	Sent (see page 983)	Indicates whether the ISAPI response message has been sent.

25.1.2.2 TISAPIResponse.Create Constructor

Creates an instance of TISAPIResponse (see page 977) to represent the response to the HTTPRequest parameter.

Pascal

```
constructor Create(HTTPRequest: TWebRequest);
```

C++

```
__fastcall TISAPIResponse(TWebRequest HTTPRequest);
```

Description

Applications don't need to call Create to instantiate a ISAPI response object. TISAPIApplication creates a TISAPIResponse (see page 977) for each TISAPIRequest (see page 970) it creates.

See Also

TISAPIApplication (see page 960)

TISAPIRequest (see page 970)

25.1.2.3 TISAPIResponse Methods

The methods of the TISAPIResponse class are listed here.

Protected Methods

	Name	Description
	GetContent (see page 979)	Reads the value of the Content property.

	GetDateVariable (see page 979)	Extracts a date from the internal table of HTTP header variables.
	GetIntegerVariable (see page 980)	Extracts an integer value from the internal table of HTTP header variables.
	GetLogMessage (see page 980)	Reads the value of the LogMessage property.
	GetStatusCode (see page 980)	Reads the value of the StatusCode property.
	GetStringVariable (see page 981)	Extracts a string value from the internal table of HTTP header variables.
	InitResponse (see page 981)	This is InitResponse, a member of class TISAPIResponse.
	SetContent (see page 983)	Sets the Content property.
	SetDateVariable (see page 983)	Sets a date-time valued HTTP header property.
	SetIntegerVariable (see page 984)	Sets an integer valued HTTP header property.
	SetLogMessage (see page 984)	Sets the value of the LogMessage property.
	SetStatusCode (see page 984)	Sets the StatusCode property.
	SetStringVariable (see page 985)	Sets a string valued HTTP header property.

Public Methods

	Name	Description
	SendRedirect (see page 982)	Sends the response with a StatusCode of 301 (Moved Permanently) and with Location set to the URI parameter.
	SendResponse (see page 982)	Sends the ISAPI response message to the web client.
	SendStream (see page 982)	Sends the ISAPI response message with the contents set to the contents of AStream.
	Sent (see page 983)	Indicates whether the ISAPI response message has been sent.

25.1.2.3.1 TISAPIResponse.GetContent Method

Reads the value of the Content property.

Pascal

```
function GetContent: AnsiString; override;
```

C++

```
virtual __fastcall AnsiString GetContent();
```

Description

GetContent is the protected access function for the Content property.

See Also

Content

SetContent (see page 983)

25.1.2.3.2 TISAPIResponse.GetDateVariable Method

Extracts a date from the internal table of HTTP header variables.

Pascal

```
function GetDateVariable(Index: Integer): TDateTime; override;
```

C++

```
virtual __fastcall TDateTime GetDateVariable(int Index);
```

Description

The ISAPI response object uses GetDateVariable to access the properties that represent dates.

See Also

Date

Expires

LastModified

SetDateVariable (see page 983)

25.1.2.3.3 TISAPIResponse.GetIntegerVariable Method

Extracts an integer value from the internal table of HTTP header variables.

Pascal

```
function GetIntegerVariable(Index: Integer): Integer; override;
```

C++

```
virtual __fastcall int GetIntegerVariable(int Index);
```

Description

The ISAPI response object uses GetIntegerVariable to access the properties that represent integers.

See Also

ContentLength

SetIntegerVariable (see page 984)

25.1.2.3.4 TISAPIResponse.GetLogMessage Method

Reads the value of the LogMessage property.

Pascal

```
function GetLogMessage: string; override;
```

C++

```
virtual __fastcall UnicodeString GetLogMessage();
```

Description

GetLogMessage is the protected access function for LogMessage property.

See Also

LogMessage

SetLogMessage (see page 984)

25.1.2.3.5 TISAPIResponse.GetStatusCode Method

Reads the value of the StatusCode property.

Pascal

```
function GetStatusCode: Integer; override;
```

C++

```
virtual __fastcall int GetStatusCode();
```

Description

GetStatusCode is the protected access function for the StatusCode property.

See Also

StatusCode

SetStatusCode

25.1.2.3.6 TISAPIResponse.GetStringVariable Method

Extracts a string value from the internal table of HTTP header variables.

Pascal

```
function GetStringVariable(Index: Integer): AnsiString; override;
```

C++

```
virtual __fastcall AnsiString GetStringVariable(int Index);
```

Description

The ISAPI response object uses GetStringVariable to access the properties that represent strings.

See Also

Allow

ContentEncoding

ContentType

ContentVersion

DerivedFrom

Location

Realm

ReasonString

Server

SetStringVariable (see page 985)

Title

Version

WWWAuthenticate

25.1.2.3.7 TISAPIResponse.InitResponse Method

Pascal

```
procedure InitResponse; virtual;
```

C++

```
virtual __fastcall InitResponse();
```

Description

This is InitResponse, a member of class TISAPIResponse.

25.1.2.3.8 TISAPIResponse.SendRedirect Method

Sends the response with a StatusCode of 301 (Moved Permanently) and with Location set to the URI parameter.

Pascal

```
procedure SendRedirect(const URI: AnsiString); override;
```

C++

```
virtual __fastcall SendRedirect(const AnsiString URI);
```

Description

Call SendRedirect to redirect the web client request to another URI that is the proper target of the request. SendRedirect does the same thing as setting the StatusCode to 301, the Location to URI, and calling SendResponse (see page 982).

See Also

Location

SendResponse (see page 982)

StatusCode

25.1.2.3.9 TISAPIResponse.SendResponse Method

Sends the ISAPI response message to the web client.

Pascal

```
procedure SendResponse; override;
```

C++

```
virtual __fastcall SendResponse();
```

Description

Call SendResponse after setting the properties of the HTTP response object to specify the response that should be made to the web client. If the request should be redirected to another URI, it is more efficient to call SendRedirect (see page 982).

See Also

SendRedirect (see page 982)

25.1.2.3.10 TISAPIResponse.SendStream Method

Sends the ISAPI response message with the contents set to the contents of AStream.

Pascal

```
procedure SendStream(AStream: TStream); override;
```

C++

```
virtual __fastcall SendStream(TStream AStream);
```

Description

Only call SendStream after first calling SendResponse (see page 982). Unlike SendStream, SendResponse (see page 982) fills the header of the HTTP response message, based on the properties of the TISAPIResponse (see page 977) object. If SendResponse (see page 982) has not been called first to set the header of the HTTP response message, set the ContentStream property to AStream, and call SendResponse (see page 982).

See Also

ContentStream

SendResponse (see page 982)

Sent (see page 983)

25.1.2.3.11 TISAPIResponse.Sent Method

Indicates whether the ISAPI response message has been sent.

Pascal

```
function Sent: Boolean; override;
```

C++

```
virtual __fastcall Boolean Sent();
```

Description

Sent returns true if the SendRedirect (see page 982) or SendStream (see page 982) method has been called. TISAPIApplication objects read Sent to determine whether they should send a response to the web client.

See Also

SendRedirect (see page 982)

SendStream (see page 982)

25.1.2.3.12 TISAPIResponse.SetContent Method

Sets the Content property.

Pascal

```
procedure SetContent(const Value: AnsiString); override;
```

C++

```
virtual __fastcall SetContent(const AnsiString Value);
```

Description

SetContent is the protected method used by ISAPI response objects to set the Content property.

See Also

Content

GetContent (see page 979)

25.1.2.3.13 TISAPIResponse.SetDateVariable Method

Sets a date-time valued HTTP header property.

Pascal

```
procedure SetDateVariable(Index: Integer; const Value: TDateTime); override;
```

C++

```
virtual __fastcall SetDateVariable(int Index, const TDateTime Value);
```

Description

SetDateVariable writes date-time to the internal table used to store HTTP header variables. The ISAPI response object uses SetDateVariable to set the properties that represent dates.

See Also

Date

Expires

GetDateVariable (see page 979)

LastModified

25.1.2.3.14 TISAPIResponse.SetIntegerVariable Method

Sets an integer valued HTTP header property.

Pascal

```
procedure SetIntegerVariable(Index: Integer; Value: Integer); override;
```

C++

```
virtual __fastcall SetIntegerVariable(int Index, int Value);
```

Description

SetIntegerVariable writes integer values to the internal table used to store HTTP header variables. The ISAPI response object uses SetIntegerVariable to set the properties that represent integers.

See Also

ContentLength

GetIntegerVariable (see page 980)

25.1.2.3.15 TISAPIResponse.SetLogMessage Method

Sets the value of the LogMessage property.

Pascal

```
procedure SetLogMessage(const Value: string); override;
```

C++

```
virtual __fastcall SetLogMessage(const UnicodeString Value);
```

Description

SetLogMessage is the protected method used by ISAPI response objects to set the LogMessage property.

See Also

GetLogMessage (see page 980)

LogMessage

25.1.2.3.16 TISAPIResponse.SetStatusCode Method

Sets the StatusCode property.

Pascal

```
procedure SetStatusCode(Value: Integer); override;
```

C++

```
virtual __fastcall SetStatusCode(int Value);
```

Description

SetStatusCode is the protected method used by ISAPI response objects to set the StatusCode property.

See Also

StatusCode

GetStatusCode

25.1.2.3.17 TISAPIResponse.SetStringVariable Method

Sets a string valued HTTP header property.

Pascal

```
procedure SetStringVariable(Index: Integer; const Value: AnsiString); override;
```

C++

```
virtual __fastcall SetStringVariable(int Index, const AnsiString Value);
```

Description

SetStringVariable writes string values to the internal table used to store HTTP header variables. The ISAPI response object uses SetStringVariable to set the properties that represent strings.

See Also

Allow

ContentEncoding

ContentType

ContentVersion

DerivedFrom

GetStringVariable (see page 981)

Location

Realm

ReasonString

Server

Title

Version

WWWAuthenticate

26 Java Namespace

This is namespace Java.

Namespaces

Name	Description
Sql (see page 988)	This is namespace Java.Sql.

26.1 Java.Sql Namespace

This is namespace Java.Sql.

Topics

Name	Description
Classes (see page 988)	The following table lists classes in this documentation.

Classes

	Name	Description
	SQLException (see page 988)	This is class Java.Sql.SQLException.

26.1.1 Classes

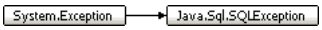
The following table lists classes in this documentation.

Classes

	Name	Description
	SQLException (see page 988)	This is class Java.Sql.SQLException.

26.1.1.1 SQLException Class

Class Hierarchy



Pascal

```
SQLException = class(System.Exception);
```

C++

```
class SQLException : public System.Exception;
```

File

SQLException.cs

Description

This is class Java.Sql.SQLException.

26.1.1.1.1 SQLException Members

The following tables list the members exposed by SQLException.

Public Methods

	Name	Description
	SQLException (see page 989)	This is the overview for the SQLException constructor overload.

Public Properties

	Name	Description
	ErrorCode (see page 989)	This is ErrorCode, a member of class SQLException.
	SQLState (see page 990)	This is SQLState, a member of class SQLException.

26.1.1.1.2 SQLException Constructor

This is the overview for the SQLException constructor overload.

Overload List

	Name	Description
	SQLException.SQLException (string) (see page 989)	This is SQLException, a member of class SQLException.
	SQLException.SQLException (string, string, int, Exception) (see page 989)	This is SQLException, a member of class SQLException.

26.1.1.1.2.1 SQLException.SQLException Constructor (string)

Pascal

```
constructor Create(reason: string);
```

C++

```
SQLException(string reason);
```

Description

This is SQLException, a member of class SQLException.

26.1.1.1.2.2 SQLException.SQLException Constructor (string, string, int, Exception)

Pascal

```
constructor Create(reason: string; SQLState: string; vendorCode: Integer; cause: Exception);
```

C++

```
SQLException(string reason, string SQLState, int vendorCode, Exception cause);
```

Description

This is SQLException, a member of class SQLException.

26.1.1.1.3 SQLException Properties

The properties of the SQLException class are listed here.

Public Properties

	Name	Description
	ErrorCode (see page 989)	This is ErrorCode, a member of class SQLException.
	SQLState (see page 990)	This is SQLState, a member of class SQLException.

26.1.1.1.3.1 SQLException.ErrorCode Property

Pascal

```
property ErrorCode: Integer;
```

C++

```
__property int ErrorCode;
```

Description

This is ErrorCode, a member of class SQLException.

26.1.1.1.3.2 SQLException.SQLState Property

Pascal

```
property SQLState: string;
```

C++

```
__property string SQLState;
```

Description

This is SQLState, a member of class SQLException.

27 jpeg Namespace

This is namespace jpeg.

Topics

Name	Description
Classes (see page 992)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 1011)	The following table lists structs, records, enums in this documentation.
Types (see page 1014)	The following table lists types in this documentation.
Variables (see page 1015)	The following table lists variables in this documentation.

Classes

	Name	Description
	TJPEGData (see page 992)	TJPEGData encapsulates the compressed file that contains the data source for the original compressed jpeg image used by a TJPEGImage (see page 994) object.
	TJPEGImage (see page 994)	TJPEGImage encapsulates a jpeg graphic.

Structs, Records, Enums

	Name	Description
	TJPEGDefaults (see page 1011)	This is record jpeg.TJPEGDefaults.
	TJPEGPerformance (see page 1012)	TJPEGPerformance represents the type of optimization to use when displaying a JPEG image.
	TJPEGPixelFormat (see page 1012)	TJPEGPixelFormat indicates the format used to display a JPEG image.
	TJPEGScale (see page 1013)	TJPEGScale represents the display size of a JPEG image.

Types

Name	Description
TJPEGQualityRange (see page 1014)	TJPEGQualityRange represents the trade-off ratio between image quality and file size in a JPEG image.

Variables

Name	Description
JPEGDefaults (see page 1015)	This is variable jpeg.JPEGDefaults.

27.1 Classes

The following table lists classes in this documentation.

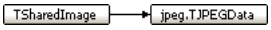
Classes

	Name	Description
	TJPEGData (see page 992)	TJPEGData encapsulates the compressed file that contains the data source for the original compressed jpeg (see page 991) image used by a TJPEGImage (see page 994) object.
	TJPEGImage (see page 994)	TJPEGImage encapsulates a jpeg (see page 991) graphic.

27.1.1 TJPEGData Class

TJPEGData encapsulates the compressed file that contains the data source for the original compressed jpeg (see page 991) image used by a TJPEGImage (see page 994) object.

Class Hierarchy



Pascal

```
TJPEGData = class(TSharedImage);
```

C++

```
class TJPEGData : public TSharedImage;
```

File

jpeg

Description

TJPEGData is used for the internal implementation of TJPEGImage (see page 994). TJPEGData is created automatically by the TJPEGImage (see page 994) object when that image object is created.

TJPEGData owns the file handle to the jpeg (see page 991) data source. Thus, TJPEGData implements how TJPEGImage (see page 994) does reference counting and handle sharing. When a jpeg (see page 991) image object is copied using the Assign method, those multiple instances of the image object refer to the same TJPEGData object.

The TJPEGData file itself is never modified. Each TJPEGImage (see page 994) instance that shares it has its own properties that determine how this data is decompressed into its own internal, local bitmap representation. However, if a TJPEGImage (see page 994) needs to modify the actual jpeg (see page 991) data, it breaks its link to that particular TJPEGData instance and creates a new one.

See Also

TJPEGImage (see page 994)

27.1.1.1 TJPEGData Members

The following tables list the members exposed by TJPEGData.

Protected Methods

	Name	Description
	FreeHandle (see page 993)	This is FreeHandle, a member of class TJPEGData.

Public Methods

	Name	Description
	Destroy (see page 993)	Destroys of an instance of a jpeg (see page 991) data object.

27.1.1.2 TJPEGData.Destroy Destructor

Destroys of an instance of a jpeg (see page 991) data object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TJPEGData();
```

Description

The jpeg (see page 991) data object is automatically destroyed by the jpeg (see page 991) image object that uses it. It is not necessary to call Destroy or to directly access the jpeg (see page 991) data object.

The Destroy method of a jpeg (see page 991) data object handles the freeing of the data.

See Also

System::TObject::Free

System::TObject::Destroy

27.1.1.3 TJPEGData Methods

The methods of the TJPEGData class are listed here.

Protected Methods

	Name	Description
	FreeHandle (see page 993)	This is FreeHandle, a member of class TJPEGData.

27.1.1.3.1 TJPEGData.FreeHandle Method

Pascal

```
procedure FreeHandle; override;
```

C++

```
virtual __fastcall FreeHandle();
```

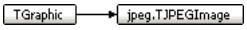
Description

This is FreeHandle, a member of class TJPEGData.

27.1.2 TJPEGImage Class

TJPEGImage encapsulates a jpeg (see page 991) graphic.

Class Hierarchy



Pascal

```
TJPEGImage = class(TGraphic);
```

C++

```
class TJPEGImage : public TGraphic;
```

File

jpeg

Description

Use TJPEGImage to read and write jpeg (see page 991) compressed image data. TJPEGImage handles the digital compression and decompression of still images for use in computer systems. It uses the data from an instance of TJPEGData (see page 992), which contains the actual jpeg (see page 991) data source and is never modified. Each jpeg (see page 991) image object may share its TJPEGData (see page 992) object with other instances of a jpeg (see page 991) image by creating copies using the Assign (see page 998) method. The jpeg (see page 991) data source handles reference counting for the jpeg (see page 991) image objects that are linked to it.

TJPEGImage has an internal bitmap that represents the jpeg (see page 991) image. This internal image and the original source of the jpeg (see page 991) image are read only. TJPEGImage has properties that determine how each instance will handle color conversion, compression, decompression, performance, and so on.

The following are characteristics of this object. A TJPEGImage object:

Has no canvas (so it cannot draw onto a canvas). However, TJPEGImage implements the protected Draw (see page 1001) method introduced in TGraphic, so it can draw itself on the canvas of another object.

Provides no access to the internal bitmap image that it creates for the JPEG image.

Performs reference counting and handle sharing by means of the TJPEGData (see page 992) object. Multiple instances can refer to the same TJPEGData (see page 992) image. TJPEGData (see page 992) is the actual owner of the file handle to the jpeg (see page 991) data source.

See Also

TJPEGData (see page 992)

TBitmap

27.1.2.1 TJPEGImage Members

The following tables list the members exposed by TJPEGImage.

Protected Methods

	Name	Description
	AssignTo (see page 999)	Copies the properties of an object to a destination object.
	Changed (see page 1000)	Called when graphic has changed.

	Draw (see page 1001)	Render graphic onto canvas at rectangle.
	Equals (see page 1001)	Compare graphic to another TGraphic object and return true if objects contain same graphic.
	FreeBitmap (see page 1001)	This is FreeBitmap, a member of class TJPEGImage.
	GetEmpty (see page 1001)	Indicates whether graphics object contains graphic.
	GetHeight (see page 1002)	Get vertical size of graphic in pixels.
	GetPalette (see page 1002)	Get color palette of graphical image.
	GetWidth (see page 1002)	Get horizontal size of graphic in pixels.
	NewBitmap (see page 1004)	This is NewBitmap, a member of class TJPEGImage.
	NewImage (see page 1004)	This is NewImage, a member of class TJPEGImage.
	ReadData (see page 1004)	Read graphic data from TStream.
	ReadStream (see page 1005)	This is ReadStream, a member of class TJPEGImage.
	SetHeight (see page 1006)	Set vertical size of graphic in pixels.
	SetPalette (see page 1006)	This is SetPalette, a member of class TJPEGImage.
	SetWidth (see page 1006)	Set horizontal size of graphic in pixels.
	WriteData (see page 1007)	Write graphic data to TStream.

Protected Properties

	Name	Description
	Bitmap (see page 1008)	This is Bitmap, a member of class TJPEGImage.

Public Methods

	Name	Description
	Assign (see page 998)	Copies the jpeg (see page 991) image object and creates a new reference to the internal data source object.
	Compress (see page 1000)	Forces the action of compression.
	Create (see page 995)	Instantiates a jpeg (see page 991) image object.
	Destroy (see page 997)	Destroys an instance of a jpeg (see page 991) image object.
	DIBNeeded (see page 1000)	Decompress the jpeg (see page 991) image into a bitmap.
	JPEGNeeded (see page 1003)	Creates a new TJPEGData (see page 992) object if the jpeg (see page 991) image only has a bitmap and a jpeg (see page 991) is needed.
	LoadFromClipboardFormat (see page 1003)	Reads in a jpeg (see page 991) image from the Clipboard into the jpeg (see page 991) image object.
	LoadFromStream (see page 1004)	Loads the jpeg (see page 991) image from a stream.
	SaveToClipboardFormat (see page 1005)	Writes a jpeg (see page 991) image to a variable in Clipboard format.
	SaveToStream (see page 1005)	Writes the jpeg (see page 991) data to a stream.

Public Properties

	Name	Description
	CompressionQuality (see page 1008)	Indicates the trade-off ratio between the image quality and the file size.
	Grayscale (see page 1008)	Determines whether the image output of a JPEG image is black and white, or color.
	Performance (see page 1008)	Controls the trade-off between color quality and speed of decompression.
	PixelFormat (see page 1009)	Determines whether the JPEG image is displayed in 24 or 8-bit format.
	ProgressiveDisplay (see page 1009)	Controls how the image is displayed when decompressing files.
	ProgressiveEncoding (see page 1010)	Determines whether an image can be progressively displayed when it is decompressed.
	Scale (see page 1010)	Determines the size of the JPEG image when it is displayed.
	Smoothing (see page 1010)	Determines whether the jpeg (see page 991) image displays in blocks or with blurred edges.

27.1.2.2 TJPEGImage.Create Constructor

Instantiates a jpeg (see page 991) image object.

Pascal

```
constructor Create; override;
```

C++

```
virtual __fastcall TJPEGImage();
```

Description

Call Create to instantiate a jpeg (see page 991) image object at run time. Create allocates memory for a jpeg (see page 991) image object and then initializes its properties. After the jpeg (see page 991) is loaded from a file, an internal bitmap is created to hold the image.

C++ Examples:

```
/*
This example converts a bitmap image to a jpeg file by using
the Assign method.  This example requires a button and two
images.  One image must have it's Picture property loaded
with a bitmap at design time.
*/

#include <Jpeg.hpp>
#include <memory>       //for STL auto_ptr class

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  //Requires "jpeg.hpp" to be included in the source file
  std::auto_ptr<TJPEGImage> jp(new TJPEGImage());
  jp->Assign(Image1->Picture->Bitmap);
  jp->SaveToFile("..\\factory.jpg");
  Image2->Stretch = True;
  Image2->Picture->LoadFromFile("..\\factory.jpg");
}
```

Delphi Examples:

```
{
This example converts a bitmap image to a jpeg file by using
the Assign method.  This example requires a button and two
images.  One image must have it's Picture property loaded
with a bitmap at design time.
}
uses Jpeg;

procedure TForm1.Button1Click(Sender: TObject);
var
  jp: TJPEGImage;  //Requires the "jpeg" unit added to "uses" clause.
begin
  jp := TJPEGImage.Create;
  try
    with jp do
    begin
      Assign(Image1.Picture.Bitmap);
      SaveToFile('factory.jpg');
      Image2.Stretch := True;
      Image2.Picture.LoadFromFile('factory.jpg');
    end;
  finally
    jp.Free;
  end;
end;
```

See Also

System::TObject::Create

27.1.2.3 TJPEGImage.Destroy Destructor

Destroys an instance of a jpeg (see page 991) image object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TJPEGImage();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TJPEGImage (see page 994) reference is not nil before it calls Destroy.

Destroy handles the freeing of the internal bitmap image and decrements the reference count on the jpeg (see page 991) data source to which it was linked.

See Also

System::TObject::Free

TJPEGData (see page 992)

27.1.2.4 TJPEGImage Methods

The methods of the TJPEGImage class are listed here.

Protected Methods

	Name	Description
	AssignTo (see page 999)	Copies the properties of an object to a destination object.
	Changed (see page 1000)	Called when graphic has changed.
	Draw (see page 1001)	Render graphic onto canvas at rectangle.
	Equals (see page 1001)	Compare graphic to another TGraphic object and return true if objects contain same graphic.
	FreeBitmap (see page 1001)	This is FreeBitmap, a member of class TJPEGImage.
	GetEmpty (see page 1001)	Indicates whether graphics object contains graphic.
	GetHeight (see page 1002)	Get vertical size of graphic in pixels.
	GetPalette (see page 1002)	Get color palette of graphical image.
	GetWidth (see page 1002)	Get horizontal size of graphic in pixels.
	NewBitmap (see page 1004)	This is NewBitmap, a member of class TJPEGImage.
	NewImage (see page 1004)	This is NewImage, a member of class TJPEGImage.
	ReadData (see page 1004)	Read graphic data from TStream.
	ReadStream (see page 1005)	This is ReadStream, a member of class TJPEGImage.
	SetHeight (see page 1006)	Set vertical size of graphic in pixels.
	SetPalette (see page 1006)	This is SetPalette, a member of class TJPEGImage.
	SetWidth (see page 1006)	Set horizontal size of graphic in pixels.
	WriteData (see page 1007)	Write graphic data to TStream.

Public Methods

	Name	Description
	Assign (see page 998)	Copies the jpeg (see page 991) image object and creates a new reference to the internal data source object.
	Compress (see page 1000)	Forces the action of compression.
	DIBNeeded (see page 1000)	Decompress the jpeg (see page 991) image into a bitmap.

	JPEGNeeded (see page 1003)	Creates a new TJPEGData (see page 992) object if the jpeg (see page 991) image only has a bitmap and a jpeg (see page 991) is needed.
V	LoadFromClipboardFormat (see page 1003)	Reads in a jpeg (see page 991) image from the Clipboard into the jpeg (see page 991) image object.
V	LoadFromStream (see page 1004)	Loads the jpeg (see page 991) image from a stream.
V	SaveToClipboardFormat (see page 1005)	Writes a jpeg (see page 991) image to a variable in Clipboard format.
V	SaveToStream (see page 1005)	Writes the jpeg (see page 991) data to a stream.

27.1.2.4.1 TJPEGImage.Assign Method

Copies the jpeg (see page 991) image object and creates a new reference to the internal data source object.

Pascal

```
procedure Assign(Source: TPersistent); override;
```

C++

```
virtual __fastcall Assign(TPersistent * Source);
```

Description

Assign copies the jpeg (see page 991) image contained in Source to the jpeg (see page 991) image object. Assign then calls the inherited Assign. Assign creates a new link to the internal jpeg (see page 991) data source by which the jpeg (see page 991) image objects do reference counting and handle sharing.

Tip: To draw on a jpeg (see page 991) image, Assign the jpeg (see page 991) image to a bitmap, and then draw on that.

Note: An object of one type can always be assigned to another object of the same type. Also, the Source can be of type TPicture if the Graphic property of the picture is a jpeg (see page 991) image or a bitmap.

C++ Examples:

```
/*
This example converts a bitmap image to a jpeg file by using
the Assign method.  This example requires a button and two
images.  One image must have it's Picture property loaded
with a bitmap at design time.
*/

#include <Jpeg.hpp>
#include <memory>       //for STL auto_ptr class

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  //Requires "jpeg.hpp" to be included in the source file
  std::auto_ptr<TJPEGImage> jp(new TJPEGImage());
  jp->Assign(Image1->Picture->Bitmap);
  jp->SaveToFile("..\\factory.jpg");
  Image2->Stretch = True;
  Image2->Picture->LoadFromFile("..\\factory.jpg");
}
```

Delphi Examples:

```
{
This example converts a bitmap image to a jpeg file by using
the Assign method.  This example requires a button and two
images.  One image must have it's Picture property loaded
with a bitmap at design time.
}
uses Jpeg;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
  jp: TJPEGImage;  //Requires the "jpeg" unit added to "uses" clause.
begin
  jp := TJPEGImage.Create;
  try
    with jp do
    begin
      Assign(Image1.Picture.Bitmap);
      SaveToFile('factory.jpg');
      Image2.Stretch := True;
      Image2.Picture.LoadFromFile('factory.jpg');
    end;
  finally
    jp.Free;
  end;
end;
```

See Also

TJPEGData (see page 992)

Assign

Graphic

27.1.2.4.2 TJPEGImage.AssignTo Method

Copies the properties of an object to a destination object.

Pascal

```
procedure AssignTo(Dest: TPersistent); override;
```

C++

```
virtual __fastcall AssignTo(TPersistent * Dest);
```

Description

Override the AssignTo method to extend the functionality of the Assign method of destination objects so that they handle newly created object classes. When defining a new object class, override the Assign method for every existing object class that should be able to copy its properties to the new class. Override the AssignTo method for every existing class to which the new class can copy.

The Assign method of TPersistent calls AssignTo if the descendant object does not succeed in copying the properties of a source object. The AssignTo method defined by TPersistent raises an EConvertError exception.

For example, given the following code in which A and B are instance variables:

```
A.Assign(B); {Delphi}
A->Assign(B); // C++
```

if A knows how to handle B, then it does so and returns. If A doesn't know how to handle B's type, execution will trickle to the TPersistent version of Assign, which calls:

```
B.AssignTo(A); {Delphi}
B->AssignTo(A); // C++
```

If B knows how to copy to A, the assignment succeeds. Otherwise, TPersistent raises an exception.

See Also

Assign

EConvertError

27.1.2.4.3 TJPEGImage.Changed Method

Called when graphic has changed.

Pascal

```
procedure Changed(Sender: TObject); override;
```

C++

```
virtual __fastcall Changed(TObject * Sender);
```

Description

Changed is called automatically whenever the TGraphic object is modified. It sets the Modified property to True and triggers the OnChange event.

See Also

Modified

OnChange

27.1.2.4.4 TJPEGImage.Compress Method

Forces the action of compression.

Pascal

```
procedure Compress;
```

C++

```
__fastcall Compress();
```

Description

Call Compress to compress for optimization, such as before streaming out when assigning a bitmap to a jpeg (see page 991). Changing properties like CompressionQuality (see page 1008) does not force compression. To force compression, call Compress.

See Also

CompressionQuality (see page 1008)

27.1.2.4.5 TJPEGImage.DIBNeeded Method

Decompress the jpeg (see page 991) image into a bitmap.

Pascal

```
procedure DIBNeeded;
```

C++

```
__fastcall DIBNeeded();
```

Description

DIBNeeded is used when the jpeg (see page 991) image needs a bitmap representation of its image. Call DIBNeeded just before drawing rather than while drawing for optimization.

See Also

JPEGNeeded (see page 1003)

27.1.2.4.6 TJPEGImage.Draw Method

Render graphic onto canvas at rectangle.

Pascal

```
procedure Draw(ACanvas: TCanvas; const Rect: TRect); override;
```

C++

```
virtual __fastcall Draw(TCanvas ACanvas, const TRect Rect);
```

Description

TCanvas.Draw calls this function to render the graphic onto its canvas at the coordinates specified by the **Rect** parameter. Descendants of TGraphic should override this function with a specific rendering implementation.

See Also

TCanvas

Draw

DrawTransparent

27.1.2.4.7 TJPEGImage.Equals Method

Compare graphic to another TGraphic object and return true if objects contain same graphic.

Pascal

```
function Equals(Graphic: TGraphic): Boolean; override;
```

C++

```
virtual __fastcall Boolean Equals(TGraphic Graphic);
```

Description

Equals compares the contents of two TGraphic objects. Equals returns false if the TGraphic are different classes or if they contain different graphical data.

27.1.2.4.8 TJPEGImage.FreeBitmap Method

Pascal

```
procedure FreeBitmap;
```

C++

```
__fastcall FreeBitmap();
```

Description

This is FreeBitmap, a member of class TJPEGImage.

27.1.2.4.9 TJPEGImage.GetEmpty Method

Indicates whether graphics object contains graphic.

Pascal

```
function GetEmpty: Boolean; override;
```

C++

```
virtual __fastcall Boolean GetEmpty();
```

Description

GetEmpty indicates whether the TGraphic object contains a graphic.

Each descendant of the TGraphic class should override the GetEmpty method to indicate whether a given TGraphic object is bound to an image.

See Also

Empty

27.1.2.4.10 TJPEGImage.GetHeight Method

Get vertical size of graphic in pixels.

Pascal

```
function GetHeight: Integer; override;
```

C++

```
virtual __fastcall int GetHeight();
```

Description

GetHeight gets the vertical size of the TGraphic object in pixels.

Each descendant of TGraphic defines its own GetHeight and SetHeight methods to access the Height property.

See Also

Height

SetHeight

27.1.2.4.11 TJPEGImage.GetPalette Method

Get color palette of graphical image.

Pascal

```
function GetPalette: HPALETTE; override;
```

C++

```
virtual __fastcall HPALETTE GetPalette();
```

Description

GetPalette gets the color palette of the graphical image. You can also use the Palette property to get the color palette of a graphical image. If the graphic does not need or use a palette, the Palette property is zero.

See Also

Palette

SetPalette

27.1.2.4.12 TJPEGImage.GetWidth Method

Get horizontal size of graphic in pixels.

Pascal

```
function GetWidth: Integer; override;
```

C++

```
virtual __fastcall int GetWidth();
```

Description

GetWidth gets the horizontal size of the graphic in pixels.

Each descendant of the TGraphic class defines its own GetWidth and SetWidth methods to access the Width property.

See Also

Width

SetWidth

27.1.2.4.13 TJPEGImage.JPEGNeeded Method

Creates a new TJPEGData (see page 992) object if the jpeg (see page 991) image only has a bitmap and a jpeg (see page 991) is needed.

Pascal

```
procedure JPEGNeeded;
```

C++

```
__fastcall JPEGNeeded();
```

Description

Call JPEGNeeded to create a new jpeg (see page 991) data source when the jpeg (see page 991) image has thrown away the jpeg (see page 991) data and only has an internal bitmap. JPEGNeeded creates a new jpeg (see page 991) data source. The SaveToStream (see page 1005) method uses JPEGNeeded.

See Also

SaveToStream

TJPEGData (see page 992)

27.1.2.4.14 TJPEGImage.LoadFromClipboardFormat Method

Reads in a jpeg (see page 991) image from the Clipboard into the jpeg (see page 991) image object.

Pascal

```
procedure LoadFromClipboardFormat(AFormat: Word; AData: THandle; APalette: HPALETTE); override;
```

C++

```
virtual __fastcall LoadFromClipboardFormat(Word AFormat, THandle AData, HPALETTE APalette);
```

Description

LoadFromClipboardFormat is called if the jpeg (see page 991) image is registered with the TPicture object using the RegisterClipboardFormat method. LoadFromClipboardFormat can also recognize, and supports, a bitmap format on the Clipboard.

LoadFromClipboardFormat replaces the current image with the data pointed to by the AData parameter. The palette for the jpeg (see page 991) image is specified by the APalette parameter.

See Also

TBitmap

TClipboard

TPicture

RegisterClipboardFormat

27.1.2.4.15 TJPEGImage.LoadFromStream Method

Loads the jpeg (see page 991) image from a stream.

Pascal

```
procedure LoadFromStream(Stream: TStream); override;
```

C++

```
virtual __fastcall LoadFromStream(TStream Stream);
```

Description

Use LoadFromStream by specifying the stream from which the jpeg (see page 991) image is loaded as the value of Stream. LoadFromStream reads a jpeg (see page 991) data source from a stream and replaces the current image with it.

See Also

SaveToStream

TJPEGData (see page 992)

27.1.2.4.16 TJPEGImage.NewBitmap Method

Pascal

```
procedure NewBitmap;
```

C++

```
__fastcall NewBitmap();
```

Description

This is NewBitmap, a member of class TJPEGImage.

27.1.2.4.17 TJPEGImage.NewImage Method

Pascal

```
procedure NewImage;
```

C++

```
__fastcall NewImage();
```

Description

This is NewImage, a member of class TJPEGImage.

27.1.2.4.18 TJPEGImage.ReadData Method

Read graphic data from TStream.

Pascal

```
procedure ReadData(Stream: TStream); override;
```

C++

```
virtual __fastcall ReadData(TStream Stream);
```

Description

ReadData is used to read the binary graphic from the virtual property **Data** (created by DefineProperties) when streaming in a TGraphic from the TStream **Stream**.

See Also

TStream

DefineProperties

WriteData

27.1.2.4.19 TJPEGImage.ReadStream Method

Pascal

```
procedure ReadStream(Size: Longint; Stream: TStream);
```

C++

```
__fastcall ReadStream(Longint Size, TStream Stream);
```

Description

This is ReadStream, a member of class TJPEGImage.

27.1.2.4.20 TJPEGImage.SaveToClipboardFormat Method

Writes a jpeg (see page 991) image to a variable in Clipboard format.

Pascal

```
procedure SaveToClipboardFormat(var AFormat: Word; var AData: THandle; var APalette:
HPALETTE); override;
```

C++

```
virtual __fastcall SaveToClipboardFormat(Word AFormat, THandle AData, HPALETTE APalette);
```

Description

Use SaveToClipboardFormat to save the jpeg (see page 991) image to a variable. The palette is returned as the APalette parameter, the format as the Aformat parameter, and the data as the AData parameter. Before the jpeg (see page 991) image can be saved, an application must have registered the format with the TJPEGImage (see page 994) object using the RegisterClipboardFormat method.

See Also

RegisterClipboardFormat

27.1.2.4.21 TJPEGImage.SaveToStream Method

Writes the jpeg (see page 991) data to a stream.

Pascal

```
procedure SaveToStream(Stream: TStream); override;
```

C++

```
virtual __fastcall SaveToStream(TStream Stream);
```

Description

Use SaveToStream to save the jpeg (see page 991) data to the stream specified by the Stream parameter. SaveToStream expects jpeg (see page 991) image data and may incur overhead of a compression cycle if the source is a bitmap.

See Also

LoadFromStream (see page 1004)

27.1.2.4.22 TJPEGImage.SetHeight Method

Set vertical size of graphic in pixels.

Pascal

```
procedure SetHeight(Value: Integer); override;
```

C++

```
virtual __fastcall SetHeight(int Value);
```

Description

SetHeight sets the vertical size of the graphic in pixels.

Each descendant of the TGraphic class defines its own GetHeight and SetHeight methods to access the Height property.

See Also

Height

GetHeight

27.1.2.4.23 TJPEGImage.SetPalette Method

Pascal

```
procedure SetPalette(Value: HPalette); override;
```

C++

```
virtual __fastcall SetPalette(HPalette Value);
```

Description

This is SetPalette, a member of class TJPEGImage.

27.1.2.4.24 TJPEGImage.SetWidth Method

Set horizontal size of graphic in pixels.

Pascal

```
procedure SetWidth(Value: Integer); override;
```

C++

```
virtual __fastcall SetWidth(int Value);
```

Description

SetWidth sets the horizontal size of the graphic in pixels.

Each descendant of the TGraphic class defines its own GetWidth and SetWidth methods to access the Width property.

See Also

Width

GetWidth

27.1.2.4.25 TJPEGImage.WriteData Method

Write graphic data to TStream.

Pascal

```
procedure WriteData(Stream: TStream); override;
```

C++

```
virtual __fastcall WriteData(TStream Stream);
```

Description

WriteData is used to write the binary graphic to the virtual property **Data** (created by DefineProperties) when streaming a TGraphic out to the TStream **Stream**.

See Also

TStream

DefineProperties

ReadData

27.1.2.5 TJPEGImage Properties

The properties of the TJPEGImage class are listed here.

Protected Properties

	Name	Description
	Bitmap (see page 1008)	This is Bitmap, a member of class TJPEGImage.

Public Properties

	Name	Description
	CompressionQuality (see page 1008)	Indicates the trade-off ratio between the image quality and the file size.
	Grayscale (see page 1008)	Determines whether the image output of a JPEG image is black and white, or color.
	Performance (see page 1008)	Controls the trade-off between color quality and speed of decompression.
	PixelFormat (see page 1009)	Determines whether the JPEG image is displayed in 24 or 8-bit format.
	ProgressiveDisplay (see page 1009)	Controls how the image is displayed when decompressing files.
	ProgressiveEncoding (see page 1010)	Determines whether an image can be progressively displayed when it is decompressed.
	Scale (see page 1010)	Determines the size of the JPEG image when it is displayed.
	Smoothing (see page 1010)	Determines whether the jpeg (see page 991) image displays in blocks or with blurred edges.

27.1.2.5.1 TJPEGImage.Bitmap Property

Pascal

```
property Bitmap: TBitmap;
```

C++

```
__property TBitmap Bitmap;
```

Description

This is Bitmap, a member of class TJPEGImage.

27.1.2.5.2 TJPEGImage.CompressionQuality Property

Indicates the trade-off ratio between the image quality and the file size.

Pascal

```
property CompressionQuality: TJPEGQualityRange;
```

C++

```
__property TJPEGQualityRange CompressionQuality;
```

Description

Use CompressionQuality to set the compression quality of the JPEG image when writing out a jpeg (see page 991) image. Higher compression results in a poorer picture quality, but a smaller file size. This property is not used for reading in files.

TJPEGQualityRange (see page 1014) is the type of the CompressionQuality property. The higher the TJPEGQualityRange (see page 1014) value (up to a maximum of 100), the better the image quality, but the larger the file size. The lower the TJPEGQualityRange (see page 1014) value (to a minimum of 1), the smaller the resulting file size, but at the expense of picture quality.

27.1.2.5.3 TJPEGImage.Grayscale Property

Determines whether the image output of a JPEG image is black and white, or color.

Pascal

```
property Grayscale: Boolean;
```

C++

```
__property Boolean Grayscale;
```

Description

Use Grayscale for reading in and writing out, that is, for decompressing and compressing, a jpeg (see page 991) image. This property affects how the image is displayed.

GrayScale is used for speed in output optimization. When Grayscale is set to true, the color is separated from luminosity. Color takes the most time to decompress, so for previewing the image, Grayscale can be set to true for speed. The output then contains 255 shades of gray.

27.1.2.5.4 TJPEGImage.Performance Property

Controls the trade-off between color quality and speed of decompression.

Pascal

```
property Performance: TJPEGPerformance;
```

C++

```
__property TJPEGPerformance Performance;
```

Description

Use Performance to set the performance of the jpeg (see page 991) image when reading, that is, for decompressing files. This property is not used for writing out files.

Performance affects the display of the internal bitmap image. This image is a 256-color bitmap. Setting Performance to jpBestSpeed may cause some dithering in the bitmap image.

27.1.2.5.5 TJPEGImage.PixelFormat Property

Determines whether the JPEG image is displayed in 24 or 8-bit format.

Pascal

```
property PixelFormat: TJPEGPixelFormat;
```

C++

```
__property TJPEGPixelFormat PixelFormat;
```

Description

Use PixelFormat to set the pixel format of the jpeg (see page 991) image to 8-bit for video drivers that cannot display 24-bit. The native format of a jpeg (see page 991) image is 24-bit. PixelFormat is used for decompression, that is, for reading in files. PixelFormat will also apply to a bitmap if the image is copied to it.

27.1.2.5.6 TJPEGImage.ProgressiveDisplay Property

Controls how the image is displayed when decompressing files.

Pascal

```
property ProgressiveDisplay: Boolean;
```

C++

```
__property Boolean ProgressiveDisplay;
```

Description

Use ProgressiveDisplay to turn on or off the incremental display of an image when reading in files. For example, if the jpeg (see page 991) data source file is on the hard drive, progressively displaying it slows down the display time: in this case setting ProgressiveDisplay to false speeds up the displaying of the image. If the file must be downloaded, set ProgressiveDisplay to true if the downloading process will take as long as the progressive displaying of the image.

ProgressiveDisplay is used only for decompression, but is affected by the manner in which the image was compressed. Images that were compressed with ProgressiveEncoding (see page 1010) set to false cannot take advantage of ProgressiveDisplay. Setting ProgressiveDisplay to true for these images has no effect.

See Also

ProgressiveEncoding (see page 1010)

Smoothing (see page 1010)

27.1.2.5.7 TJPEGImage.ProgressiveEncoding Property

Determines whether an image can be progressively displayed when it is decompressed.

Pascal

```
property ProgressiveEncoding: Boolean;
```

C++

```
__property Boolean ProgressiveEncoding;
```

Description

Use ProgressiveEncoding to compress an image when writing out files, so that it can be progressively displayed when it is decompressed. Setting the ProgressiveDisplay (see page 1009) property to true has no effect if ProgressiveEncoding was not set to true when the image was compressed.

See Also

ProgressiveDisplay (see page 1009)

27.1.2.5.8 TJPEGImage.Scale Property

Determines the size of the JPEG image when it is displayed.

Pascal

```
property Scale: TJPEGScale;
```

C++

```
__property TJPEGScale Scale;
```

Description

Use Scale to optimize performance for previewing an image. jsFullSize displays the image as maximum size, whereas jsEighth displays an eighth of the image size, but 8 times faster.

TJPEGScale (see page 1013) is the type of the Scale property.

Scale it used for reading in files, that is, for decompression.

27.1.2.5.9 TJPEGImage.Smoothing Property

Determines whether the jpeg (see page 991) image displays in blocks or with blurred edges.

Pascal

```
property Smoothing: Boolean;
```

C++

```
__property Boolean Smoothing;
```

Description

Smoothing is used if ProgressiveDisplay (see page 1009) is true, to determine whether the jpeg (see page 991) image is displayed in small blocks or with blurred edges. If Smoothing is true, the edges are blurred during progressive displaying of the image. If Smoothing is false, raw blocks of color are used to display the image.

See Also

ProgressiveDisplay (see page 1009)

27.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TJPEGPerformance (see page 1012)	TJPEGPerformance represents the type of optimization to use when displaying a JPEG image.
	TJPEGPixelFormat (see page 1012)	TJPEGPixelFormat indicates the format used to display a JPEG image.
	TJPEGScale (see page 1013)	TJPEGScale represents the display size of a JPEG image.

Records

	Name	Description
	TJPEGDefaults (see page 1011)	This is record jpeg.TJPEGDefaults.

27.2.1 jpeg.TJPEGDefaults Record

Pascal

```
TJPEGDefaults = record
  CompressionQuality: TJPEGQualityRange;
  Grayscale: Boolean;
  Performance: TJPEGPerformance;
  PixelFormat: TJPEGPixelFormat;
  ProgressiveDisplay: Boolean;
  ProgressiveEncoding: Boolean;
  Scale: TJPEGScale;
  Smoothing: Boolean;
end;
```

C++

```
struct TJPEGDefaults {
  TJPEGQualityRange CompressionQuality;
  Boolean Grayscale;
  TJPEGPerformance Performance;
  TJPEGPixelFormat PixelFormat;
  Boolean ProgressiveDisplay;
  Boolean ProgressiveEncoding;
  TJPEGScale Scale;
  Boolean Smoothing;
};
```

File

jpeg

Description

This is record jpeg.TJPEGDefaults.

27.2.2 jpeg.TJPEGPerformance Enumeration

TJPEGPerformance represents the type of optimization to use when displaying a JPEG image.

Pascal

```
TJPEGPerformance = (
  jpBestQuality,
  jpBestSpeed
);
```

C++

```
enum TJPEGPerformance {
  jpBestQuality,
  jpBestSpeed
};
```

File

jpeg

Description

The following table lists the possible values:

Value	Meaning
jpBestQuality	Optimize for image quality. This results in slower rendering of the image.
jpBestSpeed	Optimize for speed of display. This may result in some dithering in the image.

27.2.3 jpeg.TJPEGPixelFormat Enumeration

TJPEGPixelFormat indicates the format used to display a JPEG image.

Pascal

```
TJPEGPixelFormat = (
  jf24Bit,
  jf8Bit
);
```

C++

```
enum TJPEGPixelFormat {
  jf24Bit,
  jf8Bit
};
```

File

jpeg

Description

The following table lists the possible values:

Value	Meaning
jf24Bit	Use the native 24-bit jpeg (see page 991) image format.

jf8Bit	Use an 8-bit format (for video drivers that do not support the 24-bit format)

27.2.4 jpeg.TJPEGScale Enumeration

TJPEGScale represents the display size of a JPEG image.

Pascal

```
TJPEGScale = (
  jsFullSize,
  jsHalf,
  jsQuarter,
  jsEighth
);
```

C++

```
enum TJPEGScale {
  jsFullSize,
  jsHalf,
  jsQuarter,
  jsEighth
};
```

File

jpeg

Description

The following table lists the values of TJPEGScale:

Value	Meaning
jsFullSize	Displays a full-sized image.
jsHalf	Displays a half-size image in half the time it takes to display a full-sized image.
jsQuarter	Displays a quarter-size image in a quarter of the time it takes to display a full-sized image.
jsEighth	Displays an eighth-size image in an eighth of the time it takes to display a full-sized image.

27.3 Types

The following table lists types in this documentation.

Types

Name	Description
TJPEGQualityRange (see page 1014)	TJPEGQualityRange represents the trade-off ratio between image quality and file size in a JPEG image.

27.3.1 jpeg.TJPEGQualityRange Type

TJPEGQualityRange represents the trade-off ratio between image quality and file size in a JPEG image.

Pascal

```
TJPEGQualityRange = 1..100;
```

C++

```
1..100 TJPEGQualityRange;
```

File

jpeg

Description

TJPEGQualityRange represents a scale of 1 to 100, where higher values represent better image quality (but larger file size) and lower values represent smaller file sizes but with a loss of picture quality.

27.4 Variables

The following table lists variables in this documentation.

Variables

Name	Description
JPEGDefaults (see page 1015)	This is variable jpeg.JPEGDefaults.

27.4.1 jpeg.JPEGDefaults Variable

Pascal

```
JPEGDefaults: TJPEGDefaults = ( CompressionQuality: 90; Grayscale: False; Performance:
jpBestQuality; PixelFormat: jf24Bit; ProgressiveDisplay: False; ProgressiveEncoding: False;
Scale: jsFullSize; Smoothing: True; );
```

C++

```
TJPEGDefaults JPEGDefaults = ( CompressionQuality: 90; Grayscale: False; Performance:
jpBestQuality; PixelFormat: jf24Bit; ProgressiveDisplay: False; ProgressiveEncoding: False;
Scale: jsFullSize; Smoothing: True; );
```

File

jpeg

Description

This is variable jpeg.JPEGDefaults.

28 ListActns Namespace

This is namespace ListActns.

Topics

Name	Description
Classes (see page 1019)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 1074)	The following table lists structs, records, enums in this documentation.
Types (see page 1075)	The following table lists types in this documentation.

Classes

	Name	Description
	TCustomListAction (see page 1019)	TCustomListAction is the base class for action objects that provide items to client controls.
	TCustomStaticListAction (see page 1027)	This is class ListActns.TCustomStaticListAction.
	TCustomVirtualListAction (see page 1034)	This is class ListActns.TCustomVirtualListAction.
	TListActionLink (see page 1037)	This is class ListActns.TListActionLink.
	TListControlItem (see page 1041)	TListControlItem represents an item with a text value, an image, and a data pointer.
	TListControlItems (see page 1047)	TListControlItems manages a collection of TListControlItem (see page 1041) objects.
	TStaticListAction (see page 1054)	TStaticListAction provides items to client controls.
	TStaticListItems (see page 1063)	TStaticListItems represents the items that TStaticListAction (see page 1054) supplies to client controls.
	TVirtualListAction (see page 1066)	TVirtualListAction provides items to a client controls.

Structs, Records, Enums

	Name	Description
	TListItemsSortType (see page 1074)	This is record ListActns.TListItemsSortType.

Types

Name	Description
TGetItemCountEvent (see page 1075)	This is type ListActns.TGetItemCountEvent.
TGetItemEvent (see page 1075)	This is type ListActns.TGetItemEvent.
TGetVirtualItemEvent (see page 1076)	This is type ListActns.TGetVirtualItemEvent.
TItemSelectedEvent (see page 1076)	This is type ListActns.TItemSelectedEvent.
TListCompareEvent (see page 1076)	This is type ListActns.TListCompareEvent.
TListControlItemClass (see page 1077)	This is type ListActns.TListControlItemClass.

TListItemsCompare (see page 1077)	This is type ListActns.TListItemsCompare.

28.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TCustomListAction (see page 1019)	TCustomListAction is the base class for action objects that provide items to client controls.
	TCustomStaticListAction (see page 1027)	This is class ListActns.TCustomStaticListAction.
	TCustomVirtualListAction (see page 1034)	This is class ListActns.TCustomVirtualListAction.
	TListActionLink (see page 1037)	This is class ListActns.TListActionLink.
	TListControlItem (see page 1041)	TListControlItem represents an item with a text value, an image, and a data pointer.
	TListControlItems (see page 1047)	TListControlItems manages a collection of TListControlItem (see page 1041) objects.
	TStaticListAction (see page 1054)	TStaticListAction provides items to client controls.
	TStaticListItems (see page 1063)	TStaticListItems represents the items that TStaticListAction (see page 1054) supplies to client controls.
	TVirtualListAction (see page 1066)	TVirtualListAction provides items to a client controls.

28.1.1 TCustomListAction Class

TCustomListAction is the base class for action objects that provide items to client controls.

Class Hierarchy

Pascal

```
TCustomListAction = class(TCustomAction);
```

C++

```
class TCustomListAction : public TCustomAction;
```

File

ListActns

Description

TCustomListAction implements the common behavior for actions that supply the items to a list view, list box or combo box.

When an application sets the Active (see page 1024) property of a list action to true, the action fills all client list boxes and combo boxes with items. Each item may include information such as a text caption, an image, and a pointer to data.

When the user selects an item in one of the client controls, the client "executes" the list action, which causes it to generate an OnItemSelected (see page 1027) event.

When a TCustomListAction is added to an action manager, it uses the action to generate a drop-down list on a tool bar.

See Also

TStaticListAction (see page 1054)

TVirtualListAction (see page 1066)

28.1.1.1 TCustomListAction Members

The following tables list the members exposed by TCustomListAction.

Protected Events

	Name	Description
	OnGetItemCount (see page 1027)	Occurs when the list action needs to determine the number of items it supplies to client controls.
	OnItemSelected (see page 1027)	Occurs when a client control executes the action because a user selected one of the items.

Protected Methods

	Name	Description
V	GetCount (see page 1021)	Returns the value of the Count (see page 1024) property.
V	GetString (see page 1022)	Returns the value of the Strings (see page 1026) property.
V	Loaded (see page 1023)	Initializes the action after it is loaded into memory.
V	SetString (see page 1023)	Sets the value of the Strings (see page 1026) property.

Protected Properties

	Name	Description
	Images (see page 1025)	Lists the images that the list action supplies along with items.
R	Loading (see page 1026)	Indicates whether the action is currently adding items to client controls.

Public Methods

	Name	Description
V	Create (see page 1020)	Creates an instance of TCustomListAction (see page 1019).
V	ExecuteTarget (see page 1021)	Generates an OnItemSelected (see page 1027) event.
V	HandlesTarget (see page 1022)	Indicates whether the action can execute its function.

Public Properties

	Name	Description
	Active (see page 1024)	Indicates whether the list action has supplied items to its client controls.
R	Count (see page 1024)	Indicates the number of items this list action represents.
	ItemIndex (see page 1025)	Specifies the index of the selected item.
	Strings (see page 1026)	Lists the Strings that are the values of the items this action supplies.

28.1.1.2 TCustomListAction.Create Constructor

Creates an instance of TCustomListAction (see page 1019).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomListAction(TComponent * AOwner);
```

Description

Most applications do not need to call the TCustomListAction (see page 1019) constructor because actions are added at design time to an action list or action manager.

AOwner specifies the Owner for the action. This is the component that is responsible for freeing the TCustomListAction (see page 1019) instance.

See Also

Owner

28.1.1.3 TCustomListAction Methods

The methods of the TCustomListAction class are listed here.

Protected Methods

	Name	Description
	GetCount (see page 1021)	Returns the value of the Count (see page 1024) property.
	GetString (see page 1022)	Returns the value of the Strings (see page 1026) property.
	Loaded (see page 1023)	Initializes the action after it is loaded into memory.
	SetString (see page 1023)	Sets the value of the Strings (see page 1026) property.

Public Methods

	Name	Description
	ExecuteTarget (see page 1021)	Generates an OnItemSelected (see page 1027) event.
	HandlesTarget (see page 1022)	Indicates whether the action can execute its function.

28.1.1.3.1 TCustomListAction.ExecuteTarget Method

Generates an OnItemSelected (see page 1027) event.

Pascal

```
procedure ExecuteTarget(Target: TObject); override;
```

C++

```
virtual __fastcall ExecuteTarget(TObject * Target);
```

Description

ExecuteTarget is called automatically when the user selects one of the items that the list action supplies to a client list box or combo box.

Target is the control whose item was selected.

ExecuteTarget updates the ItemIndex (see page 1025) property of all client controls that descend from TCustomListControl, and then generates an OnItemSelected (see page 1027) event, where your application can respond to the user's selection.

See Also

OnItemSelected (see page 1027)

ItemIndex (see page 1025)

28.1.1.3.2 TCustomListAction.GetCount Method

Returns the value of the Count (see page 1024) property.

Pascal

```
function GetCount: Integer; virtual;
```

C++

```
virtual __fastcall int GetCount();
```

Description

GetCount is the protected read implementation of the Count (see page 1024) property.

In TCustomListAction (see page 1019), GetCount generates an OnGetItemCount (see page 1027) event to determine the number of items that the list action supplies to clients. If there is no OnGetItemCount (see page 1027) event handler, GetCount returns –1.

Override GetCount to specify the number of items in some other way. For example, TStaticListAction (see page 1054) overrides GetCount to return the number of items in its internal list of items.

See Also

Count (see page 1024)

OnGetItemCount (see page 1027)

28.1.1.3.3 TCustomListAction.GetString Method

Returns the value of the Strings (see page 1026) property.

Pascal

```
function GetString(Index: Integer): String; virtual;
```

C++

```
virtual __fastcall UnicodeString GetString(int Index);
```

Description

GetString is the protected read implementation of the Strings (see page 1026) property.

In TCustomListAction (see page 1019), GetString always returns an empty string.

Override GetString to provide a more meaningful value to the Strings (see page 1026) property. For example, TStaticListAction (see page 1054) overrides GetString to return the caption of the specified item.

See Also

Strings (see page 1026)

28.1.1.3.4 TCustomListAction.HandlesTarget Method

Indicates whether the action can execute its function.

Pascal

```
function HandlesTarget(Target: TObject): Boolean; override;
```

C++

```
virtual __fastcall Boolean HandlesTarget(TObject * Target);
```

Description

Do not call the HandlesTarget method. This method is called automatically when the user invokes an object that is linked to this action. The HandlesTarget method provides the action object with an opportunity to indicate whether it is appropriate to execute at this time with the object specified by the Target parameter as a "target".

Target is the currently active object.

HandlesTarget always returns true because the list action's ability to perform its function is not dependant on any particular target.

See Also

ExecuteTarget (see page 1021)

UpdateTarget

28.1.1.3.5 TCustomListAction.Loaded Method

Initializes the action after it is loaded into memory.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

Loaded is called automatically when the form or data module containing the action has been completely loaded into memory. TCustomListAction (see page 1019) overrides the inherited method to set Active (see page 1024) and Images (see page 1025) properties now that the other components involved (the client controls whose items are set when Active (see page 1024) becomes true and the image list component that is the value of Images (see page 1025)) are known to be loaded into memory.

See Also

Loaded

Images (see page 1025)

Active (see page 1024)

28.1.1.3.6 TCustomListAction.SetString Method

Sets the value of the Strings (see page 1026) property.

Pascal

```
procedure SetString(Index: Integer; const Value: String); virtual;
```

C++

```
virtual __fastcall SetString(int Index, const UnicodeString Value);
```

Description

SetString is the protected write implementation of the Strings (see page 1026) property.

In TCustomListAction (see page 1019), SetString does nothing.

Override SetString to provide a more meaningful value to the Strings (see page 1026) property. For example, TStaticListAction (see page 1054) overrides SetString to set the caption of the specified item.

See Also

Strings (see page 1026)

28.1.1.4 TCustomListAction Properties

The properties of the TCustomListAction class are listed here.

Protected Properties

	Name	Description
	Images (see page 1025)	Lists the images that the list action supplies along with items.
R	Loading (see page 1026)	Indicates whether the action is currently adding items to client controls.

Public Properties

	Name	Description
	Active (see page 1024)	Indicates whether the list action has supplied items to its client controls.
R	Count (see page 1024)	Indicates the number of items this list action represents.
	ItemIndex (see page 1025)	Specifies the index of the selected item.
	Strings (see page 1026)	Lists the Strings that are the values of the items this action supplies.

28.1.1.4.1 TCustomListAction.Active Property

Indicates whether the list action has supplied items to its client controls.

Pascal

```
property Active: Boolean;
```

C++

```
__property Boolean Active;
```

Description

Set Active to true to cause the list action to fill all client controls with the items. These Items are supplied by calling the GetItem method, iterating over every index from 0 to Count (see page 1024).

See Also

GetItem (see page 1035)

Items

Items

Count (see page 1024)

Items

Strings (see page 1026)

28.1.1.4.2 TCustomListAction.Count Property

Indicates the number of items this list action represents.

Pascal

```
property Count: Integer;
```

C++

```
__property int Count;
```

Description

Read Count to determine the number of items the list action supplies to client controls. This value provides an upper limit to the index that is supplied to the GetItem method or the Strings (see page 1026) property in order to fetch individual items.

In TCustomListAction (see page 1019), the value of Count is supplied by the OnGetItemCount (see page 1027) event handler. If there is no OnGetItemCount (see page 1027) event handler, the value of Count is –1. Thus, a Count of –1 indicates that Count is not defined, as opposed to 0, which indicates that there are no items.

Note: TCustomListAction (see page 1019) descendants can override the Count property access method to supply a value to Count that does not rely on an event handler. For example, TStaticListAction (see page 1054) stores all of its items in an internal list, and overrides GetCount (see page 1021) to indicate the number of items in that list.

See Also

GetItem (see page 1035)

OnGetItemCount (see page 1027)

Strings (see page 1026)

28.1.1.4.3 TCustomListAction.Images Property

Lists the images that the list action supplies along with items.

Pascal

```
property Images: TCustomImageList;
```

C++

```
__property TCustomImageList Images;
```

Description

For each item that the list action supplies to clients, it can supply an image index along with other information such as the text value. The supplied image index refers to an image in the Images property.

Note: List actions do not supply images to all clients. The Images property controls the SmallImages property of a list view or the Images property of a TComboBoxEx descendant.

See Also

SmallImages

Images

28.1.1.4.4 TCustomListAction.ItemIndex Property

Specifies the index of the selected item.

Pascal

```
property ItemIndex: Integer;
```

C++

```
__property int ItemIndex;
```

Description

ItemIndex represents the index of the selected item in all client controls that descend from TCustomListControl. Setting ItemIndex causes these clients to change their selection. When the user updates the selection using one of these clients, TCustomListAction (see page 1019) automatically updates the others so that they select the same item.

Note: List actions do not supply ItemIndex to all clients. The ItemIndex property controls only the ItemIndex property of clients that descend from TCustomListControl.

See Also

OnItemSelected (see page 1027)

ItemIndex

28.1.1.4.5 TCustomListAction.Loading Property

Indicates whether the action is currently adding items to client controls.

Pascal

```
property Loading: Boolean;
```

C++

```
__property Boolean Loading;
```

Description

Loading is used internally to detect when the list action is in the process of supplying client controls with items.

See Also

Active (see page 1024)

28.1.1.4.6 TCustomListAction.Strings Property

Lists the Strings that are the values of the items this action supplies.

Pascal

```
property Strings [Index: Integer]: String;
```

C++

```
__property UnicodeString Strings[int Index];
```

Description

Read Strings to get or set the text value for any of the items this action supplies to clients.

Index is the index of the item. It is a value from 0 to Count (see page 1024) – 1.

In TCustomListAction (see page 1019), Strings is not implemented. Reading Strings always returns an empty string, and setting Strings has no effect.

Descendant classes can override the protected GetString (see page 1022) and SetString (see page 1023) methods to provide an implementation for this property. However, there is no guarantee that Strings is implemented for any particular TCustomListAction (see page 1019) descendant.

See Also

Active (see page 1024)

Count (see page 1024)

28.1.1.5 TCustomListAction Events

The events of the TCustomListAction class are listed here.

Protected Events

	Name	Description
	OnGetItemCount (see page 1027)	Occurs when the list action needs to determine the number of items it supplies to client controls.
	OnItemSelected (see page 1027)	Occurs when a client control executes the action because a user selected one of the items.

28.1.1.5.1 TCustomListAction.OnGetItemCount Event

Occurs when the list action needs to determine the number of items it supplies to client controls.

Pascal

```
property OnGetItemCount: TGetItemCountEvent;
```

C++

```
__property TGetItemCountEvent OnGetItemCount;
```

Description

The GetCount (see page 1021) method generates an OnGetItemCount event to supply the value of the Count (see page 1024) property. Write an OnGetItemCount event handler to indicate the number of items for which you are supplying values in the OnGetItem event handler.

Sender is the list action that is supplying data to client controls.

Count (see page 1024) returns the number of items this list action supplies to client controls.

See Also

GetCount (see page 1021)

Count (see page 1024)

28.1.1.5.2 TCustomListAction.OnItemSelected Event

Occurs when a client control executes the action because a user selected one of the items.

Pascal

```
property OnItemSelected: TItemSelectedEvent;
```

C++

```
__property TItemSelectedEvent OnItemSelected;
```

Description

Write an OnItemSelected event handler to respond when the user selects one of the items supplied to client controls. Before this event is called, the list action first updates the ItemIndex (see page 1025) property of any client controls that descend from TCustomListControl.

Sender is the list action that supplies items to the control in which an item was selected.

Control is the control in which the user selected an item.

See Also

ExecuteTarget (see page 1021)

ItemIndex (see page 1025)

28.1.2 TCustomStaticListAction Class

Class Hierarchy

Pascal

```
TCustomStaticListAction = class(TCustomListAction);
```

C++

```
class TCustomStaticListAction : public TCustomListAction;
```

File

ListActns

Description

This is class ListActns.TCustomStaticListAction.

28.1.2.1 TCustomStaticListAction Members

The following tables list the members exposed by TCustomStaticListAction.

Protected Events

	Name	Description
	OnGetItemCount (see page 1027)	Occurs when the list action needs to determine the number of items it supplies to client controls.
	OnItemSelected (see page 1027)	Occurs when a client control executes the action because a user selected one of the items.

Protected Methods

	Name	Description
V	GetCount (see page 1030)	Returns the value of the Count property.
	GetItem (see page 1030)	Fills in a specified item to match the corresponding item in the Items (see page 1032) list.
V	GetItemClass (see page 1030)	Returns the class type for the items in the Items (see page 1032) property array.
V	GetString (see page 1031)	Returns the value of the Strings property.
V	Loaded (see page 1023)	Initializes the action after it is loaded into memory.
V	SetString (see page 1031)	Sets the value of the Strings property.

Protected Properties

	Name	Description
	Clients (see page 1032)	This is Clients, a member of class TCustomStaticListAction.
	Images (see page 1025)	Lists the images that the list action supplies along with items.
R	Loading (see page 1026)	Indicates whether the action is currently adding items to client controls.

Public Events

	Name	Description
	OnGetItem (see page 1033)	Occurs when the static list action supplies an item to its client controls.

Public Methods

	Name	Description
V	Create (see page 1029)	Creates an instance of TCustomStaticListAction (see page 1027).
V	Destroy (see page 1029)	Frees the memory associated with the TCustomStaticListAction (see page 1027) object.
V	ExecuteTarget (see page 1021)	Generates an OnItemSelected (see page 1027) event.
V	HandlesTarget (see page 1022)	Indicates whether the action can execute its function.

Public Properties

	Name	Description
	Active (see page 1024)	Indicates whether the list action has supplied items to its client controls.

	Count (see page 1032)	Indicates the number of items this list action represents.
	ItemIndex (see page 1025)	Specifies the index of the selected item.
	Items (see page 1032)	Lists the items that this list action supplies to clients.
	Strings (see page 1026)	Lists the Strings that are the values of the items this action supplies.

28.1.2.2 TCustomStaticListAction.Create Constructor

Creates an instance of TCustomStaticListAction (see page 1027).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomStaticListAction(TComponent * AOwner);
```

Description

Most applications do not need to call the TCustomStaticListAction (see page 1027) constructor, because actions are added at design time to an action list or action manager. `AOwner` specifies the owner for the action. This is the component that is responsible for freeing the TCustomStaticListAction (see page 1027) instance.

See Also

Count (see page 1032)

Strings (see page 1026)

28.1.2.3 TCustomStaticListAction.Destroy Destructor

Frees the memory associated with the TCustomStaticListAction (see page 1027) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomStaticListAction();
```

Description

Do not explicitly destroy actions. Action objects are created and destroyed by the action list or action manager that uses them.

See Also

Create (see page 1029)

28.1.2.4 TCustomStaticListAction Methods

The methods of the TCustomStaticListAction class are listed here.

Protected Methods

	Name	Description
	GetCount (see page 1030)	Returns the value of the Count property.
	GetItem (see page 1030)	Fills in a specified item to match the corresponding item in the Items (see page 1032) list.
	GetItemClass (see page 1030)	Returns the class type for the items in the Items (see page 1032) property array.

	GetString (see page 1031)	Returns the value of the Strings property.
	SetString (see page 1031)	Sets the value of the Strings property.

28.1.2.4.1 TCustomStaticListAction.GetCount Method

Returns the value of the Count property.

Pascal

```
function GetCount: Integer; override;
```

C++

```
virtual __fastcall int GetCount();
```

Description

GetCount is the protected read implementation of the Count property.

In TCustomListAction, GetCount generates an OnGetItemCount event to determine the number of items that the list action supplies to clients. If there is no OnGetItemCount event handler, GetCount returns –1.

Override GetCount to specify the number of items in some other way. For example, TStaticListAction overrides GetCount to return the number of items in its internal list of items.

See Also

Count (see page 1024)

OnGetItemCount (see page 1027)

28.1.2.4.2 TCustomStaticListAction.GetItem Method

Fills in a specified item to match the corresponding item in the Items (see page 1032) list.

Pascal

```
function GetItem(const Index: Integer; AnItem: TListControlItem): Boolean;
```

C++

```
__fastcall Boolean GetItem(const int Index, TListControlItem AnItem);
```

Description

When you set the Active (see page 1024) property to True, the static list action supplies items to client controls through a series of calls to GetItem, one for each item. `Index` indicates which item is desired, where 0 indicates the first item, 1 indicates the second item and so on up to the number of items specified by the Count (see page 1032) property. `AnItem` is the item supplied to clients. GetItem assigns the properties of the corresponding item in the Items (see page 1032) property to `AnItem`.

See Also

GetItemClass (see page 1030)

28.1.2.4.3 TCustomStaticListAction.GetItemClass Method

Returns the class type for the items in the Items (see page 1032) property array.

Pascal

```
function GetItemClass: TListControlItemClass; virtual;
```

C++

```
virtual __fastcall TListControlItemClass GetItemClass();
```

Description

GetItemClass returns the class type used for the items that TCustomStaticListAction (see page 1027) supplies to clients. This class type determines the item properties that can be assigned to the various client items. As implemented in TCustomStaticListAction (see page 1027), GetItemClass returns TListControlItem (see page 1041).

See Also

GetItem (see page 1030)

28.1.2.4.4 TCustomStaticListAction.GetString Method

Returns the value of the Strings property.

Pascal

```
function GetString(Index: Integer): String; override;
```

C++

```
virtual __fastcall UnicodeString GetString(int Index);
```

Description

GetString is the protected read implementation of the Strings property.

In TCustomListAction, GetString always returns an empty string.

Override GetString to provide a more meaningful value to the Strings property. For example, TStaticListAction overrides GetString to return the caption of the specified item.

See Also

Strings (see page 1026)

28.1.2.4.5 TCustomStaticListAction.SetString Method

Sets the value of the Strings property.

Pascal

```
procedure SetString(Index: Integer; const Value: String); override;
```

C++

```
virtual __fastcall SetString(int Index, const UnicodeString Value);
```

Description

SetString is the protected write implementation of the Strings property.

In TCustomListAction, SetString does nothing.

Override SetString to provide a more meaningful value to the Strings property. For example, TStaticListAction overrides SetString to set the caption of the specified item.

See Also

Strings (see page 1026)

28.1.2.5 TCustomStaticListAction Properties

The properties of the TCustomStaticListAction class are listed here.

Protected Properties

	Name	Description
	Clients (see page 1032)	This is Clients, a member of class TCustomStaticListAction.

Public Properties

	Name	Description
	Count (see page 1032)	Indicates the number of items this list action represents.
	Items (see page 1032)	Lists the items that this list action supplies to clients.

28.1.2.5.1 TCustomStaticListAction.Clients Property

Pascal

```
property Clients;
```

C++

```
__property Clients;
```

Description

This is Clients, a member of class TCustomStaticListAction.

28.1.2.5.2 TCustomStaticListAction.Count Property

Indicates the number of items this list action represents.

Pascal

```
property Count: Integer;
```

C++

```
__property int Count;
```

Description

Read Count to determine the number of items the list action supplies to client controls. This value provides an upper limit to the index that is supplied to the GetItem method or the Strings property in order to fetch individual items.

In TCustomListAction, the value of Count is supplied by the OnGetItemCount event handler. If there is no OnGetItemCount event handler, the value of Count is –1. Thus, a Count of –1 indicates that Count is not defined, as opposed to 0, which indicates that there are no items.

Note: TCustomListAction descendants can override the Count property access method to supply a value to Count that does not rely on an event handler. For example, TStaticListAction stores all of its items in an internal list, and overrides GetCount to indicate the number of items in that list.

See Also

GetItem (see page 1035)

OnGetItemCount (see page 1027)

Strings (see page 1026)

28.1.2.5.3 TCustomStaticListAction.Items Property

Lists the items that this list action supplies to clients.

Pascal

```
property Items: TStaticListItems;
```

C++

```
__property TStaticListItems Items;
```

Description

Set Items to indicate the items that this action supplies to clients. At design time, you can use the collection editor to add items to the list and set their properties.

See Also

Count (see page 1032)

Strings (see page 1026)

28.1.2.6 TCustomStaticListAction Events

The events of the TCustomStaticListAction class are listed here.

Public Events

	Name	Description
	OnGetItem (see page 1033)	Occurs when the static list action supplies an item to its client controls.

28.1.2.6.1 TCustomStaticListAction.OnGetItem Event

Occurs when the static list action supplies an item to its client controls.

Pascal

```
property OnGetItem: TGetItemEvent;
```

C++

```
__property TGetItemEvent OnGetItem;
```

Description

Write an OnGetItem event handler to customize the properties of an item before it is supplied to client controls. The GetItem (see page 1030) method generates an OnGetItem event for every item it supplies to clients, passing in the current item from the Items (see page 1032) property. You can change that item's properties in the event handler before it is added to client controls.

Sender is the list action that is supplying data to client controls.

Index indicates which item is requested, where 0 is the first item, 1 is the second item and so on up to the index that is one less than the value of the Count (see page 1032) property.

Item is the item that the action is about to supply to the user. You can change any of its properties before it is passed on to the action's client.

See Also

Count (see page 1032)

Items (see page 1059)

Images (see page 1025)

Objects

28.1.3 TCustomVirtualListAction Class

Class Hierarchy

Pascal

```
TCustomVirtualListAction = class(TCustomListAction);
```

C++

```
class TCustomVirtualListAction : public TCustomListAction;
```

File

ListActns

Description

This is class ListActns.TCustomVirtualListAction.

28.1.3.1 TCustomVirtualListAction Members

The following tables list the members exposed by TCustomVirtualListAction.

Protected Events

	Name	Description
	OnGetItemCount (see page 1027)	Occurs when the list action needs to determine the number of items it supplies to client controls.
	OnItemSelected (see page 1027)	Occurs when a client control executes the action because a user selected one of the items.

Protected Methods

	Name	Description
V	GetCount (see page 1021)	Returns the value of the Count (see page 1024) property.
	GetItem (see page 1035)	Provides information about a specified item.
V	GetString (see page 1035)	Returns the value of the Strings property.
V	Loaded (see page 1023)	Initializes the action after it is loaded into memory.
V	SetString (see page 1023)	Sets the value of the Strings (see page 1026) property.

Protected Properties

	Name	Description
	Images (see page 1025)	Lists the images that the list action supplies along with items.
R	Loading (see page 1026)	Indicates whether the action is currently adding items to client controls.

Public Events

	Name	Description
	OnGetItem (see page 1037)	Occurs when the list action needs to supply an item to its client controls.

Public Methods

	Name	Description
V	Create (see page 1020)	Creates an instance of TCustomListAction (see page 1019).
V	ExecuteTarget (see page 1021)	Generates an OnItemSelected (see page 1027) event.
V	HandlesTarget (see page 1022)	Indicates whether the action can execute its function.

Public Properties

	Name	Description
	Active (see page 1024)	Indicates whether the list action has supplied items to its client controls.
	Count (see page 1036)	Indicates the number of items this list action represents.
	ItemIndex (see page 1025)	Specifies the index of the selected item.
	Strings (see page 1026)	Lists the Strings that are the values of the items this action supplies.

28.1.3.2 TCustomVirtualListAction Methods

The methods of the TCustomVirtualListAction class are listed here.

Protected Methods

	Name	Description
	GetItem (see page 1035)	Provides information about a specified item.
V	GetString (see page 1035)	Returns the value of the Strings property.

28.1.3.2.1 TCustomVirtualListAction.GetItem Method

Provides information about a specified item.

Pascal

```
function GetItem(const Index: Integer; var Value: String; var ImageIndex: Integer; var Data:
TCustomData): Boolean;
```

C++

```
__fastcall Boolean GetItem(const int Index, UnicodeString Value, int ImageIndex, TCustomData
Data);
```

Description

When you set the Active (see page 1024) property to True, the virtual list action supplies items to client controls through a series of calls to GetItem, one for each item. GetItem collects the information about each item by generating an OnGetItem (see page 1037) event. It returns True if there is an OnGetItem (see page 1037) event handler to provide information about the item, which is returned using the GetItem parameters. GetItem returns False if it cannot supply information about the item because there is no OnGetItem (see page 1037) event handler. `Index` indicates which item is desired, where 0 indicates the first item, 1 indicates the second item and so on up to the number of items specified by the Count (see page 1036) property. Value returns the string value of the item. This is the text that appears in the client combo box or list box. ImageIndex returns the index of an image associated with the item. If the item is not associated with an image, ImageIndex returns –1. Otherwise, it returns the index of an image in the Images (see page 1025) property. Data returns a pointer to application-defined data that is associated with the item. For client controls that store items in a TStrings object, Data supplies the value of the Objects property for the item.

See Also

Count (see page 1024)

OnGetItemCount (see page 1027)

28.1.3.2.2 TCustomVirtualListAction.GetString Method

Returns the value of the Strings property.

Pascal

```
function GetString(Index: Integer): String; override;
```

C++

```
virtual __fastcall UnicodeString GetString(int Index);
```

Description

GetString is the protected read implementation of the Strings property.

In TCustomListAction, GetString always returns an empty string.

Override GetString to provide a more meaningful value to the Strings property. For example, TStaticListAction overrides GetString to return the caption of the specified item.

See Also

Strings (see page 1026)

28.1.3.3 TCustomVirtualListAction Properties

The properties of the TCustomVirtualListAction class are listed here.

Public Properties

	Name	Description
	Count (see page 1036)	Indicates the number of items this list action represents.

28.1.3.3.1 TCustomVirtualListAction.Count Property

Indicates the number of items this list action represents.

Pascal

```
property Count: Integer;
```

C++

```
__property int Count;
```

Description

Read Count to determine the number of items the list action supplies to client controls. This value provides an upper limit to the index that is supplied to the GetItem method or the Strings property in order to fetch individual items.

In TCustomListAction, the value of Count is supplied by the OnGetItemCount event handler. If there is no OnGetItemCount event handler, the value of Count is –1. Thus, a Count of –1 indicates that Count is not defined, as opposed to 0, which indicates that there are no items.

Note: TCustomListAction descendants can override the Count property access method to supply a value to Count that does not rely on an event handler. For example, TStaticListAction stores all of its items in an internal list, and overrides GetCount to indicate the number of items in that list.

See Also

GetItem (see page 1035)

OnGetItemCount (see page 1027)

Strings (see page 1026)

28.1.3.4 TCustomVirtualListAction Events

The events of the TCustomVirtualListAction class are listed here.

Public Events

	Name	Description
	OnGetItem (see page 1037)	Occurs when the list action needs to supply an item to its client controls.

28.1.3.4.1 TCustomVirtualListAction.OnGetItem Event

Occurs when the list action needs to supply an item to its client controls.

Pascal

```
property OnGetItem: TGetVirtualItemEvent;
```

C++

```
__property TGetVirtualItemEvent OnGetItem;
```

Description

Write an OnGetItem event handler to provide the items that the virtual list action supplies to clients. When you set the Active (see page 1024) property to True, the virtual list action generates an OnGetItem event for every index from 0 to Count (see page 1036) – 1. In the event handler, supply the string value, image, and data that corresponds to the index specified by Index.

Sender is the list action that is supplying data to client controls.

Index indicates which item is requested, where 0 is the first item, 1 is the second item and so on up to the index that is one less than the value of the Count (see page 1036) property.

Value returns the string value of the item. This is the text that appears in the client combo box or list box.

ImageIndex is the index of an image associated with the item. If the item is not associated with an image, ImageIndex should return –1. Otherwise, it should return the index of the desired image in the Images (see page 1025) property.

Data returns a pointer to application-defined data that is associated with the item. For client controls that store items in a TStrings object, Data is the value of the Objects property for the item.

See Also

Count (see page 1024)

Images (see page 1025)

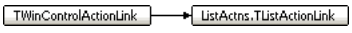
Objects

OnGetItemCount (see page 1027)

Active (see page 1024)

28.1.4 TListActionLink Class

Class Hierarchy



Pascal

```
TListActionLink = class(TWinControlActionLink);
```

C++

```
class TListActionLink : public TWinControlActionLink;
```

File

ListActns

Description

This is class ListActns.TListActionLink.

28.1.4.1 TListActionLink Members

The following tables list the members exposed by TListActionLink.

Protected Methods

	Name	Description
	AddItem (see page 1038)	This is the overview for the AddItem method overload.
	IsActiveLinked (see page 1039)	This is IsActiveLinked, a member of class TListActionLink.
	IsImagesLinked (see page 1039)	This is IsImagesLinked, a member of class TListActionLink.
	RefreshControl (see page 1040)	This is RefreshControl, a member of class TListActionLink.
	SetAction (see page 1040)	SetAction associates an action to the link's clients.
	SetActive (see page 1040)	This is SetActive, a member of class TListActionLink.
	SetImages (see page 1040)	This is SetImages, a member of class TListActionLink.
	SetItemIndex (see page 1041)	This is SetItemIndex, a member of class TListActionLink.

28.1.4.2 TListActionLink Methods

The methods of the TListActionLink class are listed here.

Protected Methods

	Name	Description
	AddItem (see page 1038)	This is the overview for the AddItem method overload.
	IsActiveLinked (see page 1039)	This is IsActiveLinked, a member of class TListActionLink.
	IsImagesLinked (see page 1039)	This is IsImagesLinked, a member of class TListActionLink.
	RefreshControl (see page 1040)	This is RefreshControl, a member of class TListActionLink.
	SetAction (see page 1040)	SetAction associates an action to the link's clients.
	SetActive (see page 1040)	This is SetActive, a member of class TListActionLink.
	SetImages (see page 1040)	This is SetImages, a member of class TListActionLink.
	SetItemIndex (see page 1041)	This is SetItemIndex, a member of class TListActionLink.

28.1.4.2.1 AddItem Method

This is the overview for the AddItem method overload.

Overload List

	Name	Description
	TListActionLink.AddItem (String, Integer, TCustomData) (see page 1039)	This is AddItem, a member of class TListActionLink.

	TListActionLink.AddItem (TListControlItem) (see page 1039)	This is AddItem, a member of class TListActionLink.

28.1.4.2.1.1 TListActionLink.AddItem Method (String, Integer, TCustomData)

Pascal

```
procedure AddItem(ACaption: String; AImageIndex: Integer; DataPtr: TCustomData); virtual;
overload;
```

C++

```
virtual __fastcall AddItem(UnicodeString ACaption, int AImageIndex, TCustomData DataPtr);
```

Description

This is AddItem, a member of class TListActionLink.

28.1.4.2.1.2 TListActionLink.AddItem Method (TListControlItem)

Pascal

```
procedure AddItem(AnItem: TListControlItem); virtual; overload;
```

C++

```
virtual __fastcall AddItem(TListControlItem AnItem);
```

Description

This is AddItem, a member of class TListActionLink.

28.1.4.2.2 TListActionLink.IsActiveLinked Method

Pascal

```
function IsActiveLinked: Boolean; virtual;
```

C++

```
virtual __fastcall Boolean IsActiveLinked();
```

Description

This is IsActiveLinked, a member of class TListActionLink.

28.1.4.2.3 TListActionLink.IsImagesLinked Method

Pascal

```
function IsImagesLinked: Boolean; virtual;
```

C++

```
virtual __fastcall Boolean IsImagesLinked();
```

Description

This is IsImagesLinked, a member of class TListActionLink.

28.1.4.2.4 TListActionLink.RefreshControl Method

Pascal

```
procedure RefreshControl;
```

C++

```
__fastcall RefreshControl();
```

Description

This is RefreshControl, a member of class TListActionLink.

28.1.4.2.5 TListActionLink.SetAction Method

SetAction associates an action to the link's clients.

Pascal

```
procedure SetAction(Value: TBasicAction); override;
```

C++

```
virtual __fastcall SetAction(TBasicAction Value);
```

Description

SetAction is the setter method for the Action property. SetAction connects an action associated with a link to the link's clients.

See Also

SetAction

Action

28.1.4.2.6 TListActionLink.SetActive Method

Pascal

```
procedure SetActive(const Value: Boolean); virtual;
```

C++

```
virtual __fastcall SetActive(const Boolean Value);
```

Description

This is SetActive, a member of class TListActionLink.

28.1.4.2.7 TListActionLink.SetImages Method

Pascal

```
procedure SetImages(Value: TCustomImageList); virtual;
```

C++

```
virtual __fastcall SetImages(TCustomImageList Value);
```

Description

This is SetImages, a member of class TListActionLink.

28.1.4.2.8 TListActionLink.SetItemIndex Method

Pascal

```
procedure SetItemIndex(const Value: Integer); virtual;
```

C++

```
virtual __fastcall SetItemIndex(const int Value);
```

Description

This is SetItemIndex, a member of class TListActionLink.

28.1.5 TListControlItem Class

TListControlItem represents an item with a text value, an image, and a data pointer.

Class Hierarchy

Pascal

```
TListControlItem = class(TCollectionItem);
```

C++

```
class TListControlItem : public TCollectionItem;
```

File

ListActns

Description

TListControlItem represents a single item in a control that lists items. TListControlItem objects are used by TStaticListAction (see page 1054) to represent the items in a variety of client controls such as list views, list boxes, and combo boxes. In addition, TListControlItem is the base class for TComboExItem, which is specialized to work with the TComboBoxEx control.

Each TListControlItem has

A text value, which it stores as the Caption (see page 1046) property.

An associated image, which is identified by the ImageIndex (see page 1047) property.

A data pointer, which it stores as the Data (see page 1047) property.

See Also

TListControlItems (see page 1047)

TComboExItem

28.1.5.1 TListControlItem Members

The following tables list the members exposed by TListControlItem.

Protected Fields

	Name	Description
	FCaption (see page 1043)	This is FCaption, a member of class TListControlItem.
	FData (see page 1043)	This is FData, a member of class TListControlItem.
	FImageIndex (see page 1043)	This is FImageIndex, a member of class TListControlItem.

Protected Methods

	Name	Description
	Changed (see page 1044)	Responds when any of the item's properties changes.
	GetDisplayName (see page 1045)	Specifies the name of the item as it appears in the Object Inspector.
	SetCaption (see page 1045)	Sets the value of the Caption (see page 1046) property.
	SetData (see page 1045)	Sets the value of the Data (see page 1047) property.
	SetImageIndex (see page 1046)	Sets the value of the ImageIndex (see page 1047) property.

Public Methods

	Name	Description
	Assign (see page 1044)	Copies the contents of another, similar object.
	Create (see page 1042)	Creates and initializes a TListControlItem (see page 1041) instance.

Public Properties

	Name	Description
	Data (see page 1047)	Points to user-defined data associated with the control.

Published Properties

	Name	Description
	Caption (see page 1046)	Specifies the text value of the item.
	ImageIndex (see page 1047)	Identifies the image associated with this item.

28.1.5.2 TListControlItem.Create Constructor

Creates and initializes a TListControlItem (see page 1041) instance.

Pascal

```
constructor Create(Collection: TCollection); override;
```

C++

```
virtual __fastcall TListControlItem(TCollection Collection);
```

Description

Do not call the constructor to create a TListControlItem (see page 1041) instance. Instead, use the collection's Add method to create a new list control item.

Collection is the collection object that owns and manages this list control item.

See Also

Add

28.1.5.3 TListControlItem Fields

The fields of the TListControlItem class are listed here.

Protected Fields

	Name	Description
	FCaption (see page 1043)	This is FCaption, a member of class TListControlItem.
	FData (see page 1043)	This is FData, a member of class TListControlItem.
	FImageIndex (see page 1043)	This is FImageIndex, a member of class TListControlItem.

28.1.5.3.1 TListControlItem.FCaption Field

Pascal

```
FCaption: String;
```

C++

```
UnicodeString FCaption;
```

Description

This is FCaption, a member of class TListControlItem.

28.1.5.3.2 TListControlItem.FData Field

Pascal

```
FData: TCustomData;
```

C++

```
TCustomData FData;
```

Description

This is FData, a member of class TListControlItem.

28.1.5.3.3 TListControlItem.FImageIndex Field

Pascal

```
FImageIndex: TImageIndex;
```

C++

```
TImageIndex FImageIndex;
```

Description

This is FImageIndex, a member of class TListControlItem.

28.1.5.4 TListControlItem Methods

The methods of the TListControlItem class are listed here.

Protected Methods

	Name	Description
	Changed (see page 1044)	Responds when any of the item's properties changes.
V	GetDisplayName (see page 1045)	Specifies the name of the item as it appears in the Object Inspector.
V	SetCaption (see page 1045)	Sets the value of the Caption (see page 1046) property.
V	SetData (see page 1045)	Sets the value of the Data (see page 1047) property.
V	SetImageIndex (see page 1046)	Sets the value of the ImageIndex (see page 1047) property.

Public Methods

	Name	Description
	Assign (see page 1044)	Copies the contents of another, similar object.

28.1.5.4.1 TListControlItem.Assign Method

Copies the contents of another, similar object.

Pascal

```
procedure Assign(Source: TPersistent); override;
```

C++

```
virtual __fastcall Assign(TPersistent * Source);
```

Description

Call Assign to copy the properties or other attributes of one object from another. The standard form of a call to Assign is

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

which tells the Destination object to copy the contents of the Source object to itself.

Most objects override Assign to handle the assignment of properties from similar objects. When overriding Assign, call the inherited method if the destination object can't handle the assignment of properties from the class of the Source parameter.

If no overridden Assign method can handle the assignment of properties from Source, the method implemented in TPersistent calls the source object's AssignTo method. This allows the source object to handle the assignment. If the Source object is nil (Delphi) or NULL (C++), Assign raises an EConvertError exception.

In general, the statement

```
Destination := Source; {Delphi}
Destination = Source; // C++
```

is not the same as the statement

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

The assignment operator makes Destination reference the same object as Source, whereas the Assign method copies the contents of the object referenced by Source into the object referenced by Destination.

Note: The types of some properties are also objects. If these properties have write methods that use Assign to set the value of the property, then in these cases the assignment operator does the same thing as the Assign method.

See Also

AssignTo

EConvertError

28.1.5.4.2 TListControlItem.Changed Method

Responds when any of the item's properties changes.

Pascal

```
procedure Changed;
```

C++

```
__fastcall Changed();
```

Description

Changed is called automatically when the item's properties change. It calls the Update method of the corresponding TListControlItems (see page 1047) object, so that the collection can perform any necessary adjustments.

See Also

Update

28.1.5.4.3 TListControlItem.GetDisplayName Method

Specifies the name of the item as it appears in the Object Inspector.

Pascal

```
function GetDisplayName: String; override;
```

C++

```
virtual __fastcall UnicodeString GetDisplayName();
```

Description

GetDisplayName returns the string that represents the item in the Collection editor and the Object Inspector.

GetDisplayName returns the value of the Caption (see page 1046) if it is set. Otherwise, it returns the class name of the TListControlItem (see page 1041) descendant for which this is an instance, with a number added to ensure uniqueness.

See Also

Caption (see page 1046)

28.1.5.4.4 TListControlItem.SetCaption Method

Sets the value of the Caption (see page 1046) property.

Pascal

```
procedure SetCaption(const Value: String); virtual;
```

C++

```
virtual __fastcall SetCaption(const UnicodeString Value);
```

Description

SetCaption is the protected write implementation of the Caption (see page 1046) property. It sets the Caption (see page 1046) property to Value, and then calls the Changed (see page 1044) method so that the collection can update itself.

See Also

Caption (see page 1046)

Changed (see page 1044)

28.1.5.4.5 TListControlItem.SetData Method

Sets the value of the Data (see page 1047) property.

Pascal

```
procedure SetData(const Value: TCustomData); virtual;
```

C++

```
virtual __fastcall SetData(const TCustomData Value);
```

Description

SetData is the protected write implementation of the Data (see page 1047) property. It sets the Data (see page 1047) property to Value, and then calls the Changed (see page 1044) method so that the collection can update itself.

See Also

Data (see page 1047)

Changed (see page 1044)

28.1.5.4.6 TListControlItem.SetImageIndex Method

Sets the value of the ImageIndex (see page 1047) property.

Pascal

```
procedure SetImageIndex(const Value: TImageIndex); virtual;
```

C++

```
virtual __fastcall SetImageIndex(const TImageIndex Value);
```

Description

SetImageIndex is the protected write implementation of the ImageIndex (see page 1047) property. It sets the ImageIndex (see page 1047) property to Value, and then calls the Changed (see page 1044) method so that the collection can update itself.

See Also

ImageIndex (see page 1047)

Changed (see page 1044)

28.1.5.5 TListControlItem Properties

The properties of the TListControlItem class are listed here.

Public Properties

	Name	Description
	Data (see page 1047)	Points to user-defined data associated with the control.

Published Properties

	Name	Description
	Caption (see page 1046)	Specifies the text value of the item.
	ImageIndex (see page 1047)	Identifies the image associated with this item.

28.1.5.5.1 TListControlItem.Caption Property

Specifies the text value of the item.

Pascal

```
property Caption: String;
```

C++

```
__property UnicodeString Caption;
```

Description

Set Caption to provide the item with a text value. This is the value that appears in list boxes or combo boxes, or which labels items in list view controls.

See Also

ImageIndex (see page 1047)

Data (see page 1047)

28.1.5.5.2 TListControlItem.Data Property

Points to user-defined data associated with the control.

Pascal

```
property Data: TCustomData;
```

C++

```
__property TCustomData Data;
```

Description

Set Data to associate this item with application-specific data. Data has no predefined meaning, although it is used for sorting items when the managing collection's SortType property is stData.

See Also

SortType (see page 1053)

ImageIndex (see page 1047)

Caption (see page 1046)

28.1.5.5.3 TListControlItem.ImageIndex Property

Identifies the image associated with this item.

Pascal

```
property ImageIndex: TImageIndex;
```

C++

```
__property TImageIndex ImageIndex;
```

Description

Set ImageIndex to associate this item with an image. ImageIndex is an index into the Images property of the component that uses this TListControlItem (see page 1041) object to represent one of its items.

See Also

Images (see page 1025)

Images

28.1.6 TListControlItems Class

TListControlItems manages a collection of TListControlItem (see page 1041) objects.

Class Hierarchy

Pascal

```
TListControlItems = class(TOwnedCollection);
```

C++

```
class TListControlItems : public TOwnedCollection;
```

File

ListActns

Description

TListControlItems manages a collection of items that each include a text value, associated image, and a data pointer.

Use the properties and methods of TListControlItems to:

Access a specific item.

Add (see page 1049) or delete items from the collection.

Find out how many items are in the collection.

Sort (see page 1051) the items in the collection.

Although TListControlItems can be used on its own, it also acts as a base class to classes that manage the items for a specific component. TListControlItems descendants include TComboExItems, which manages the items of a TComboBoxEx control, and TStaticListItems (see page 1063), which manages items for a list action component.

See Also

TListControlItem (see page 1041)

TStaticListItems (see page 1063)

TComboExItems

28.1.6.1 TListControlItems Members

The following tables list the members exposed by TListControlItems.

Protected Methods

	Name	Description
	CompareItems (see page 1050)	Returns the relative order of two items in the collection.
	Update (see page 1051)	Updates the collection to reflect changes to its items.

Public Methods

	Name	Description
	Add (see page 1049)	Adds a new item to the collection.
	Create (see page 1049)	Creates and initializes a TOwnedCollection instance.
	CustomSort (see page 1050)	Sorts the items in the collection using a specified comparison function.
	Sort (see page 1051)	Sorts the items in the collection.

Public Properties

	Name	Description
	Items (see page 1052)	Lists the items in the collection.

Published Events

	Name	Description
	OnCompare (see page 1054)	Occurs to when two items need to be compared for sorting the collection.

Published Properties

	Name	Description
	CaseSensitive (see page 1052)	Controls whether items are sorted in a case-sensitive or case-insensitive manner.
	SortType (see page 1053)	Indicates how the items should be sorted.

28.1.6.2 TListControlItems.Create Constructor

Creates and initializes a TOwnedCollection instance.

Pascal

```
constructor Create(AOwner: TPersistent; ItemClass: TCollectionItemClass); virtual;
```

C++

```
virtual __fastcall TListControlItems(TPersistent * AOwner, TCollectionItemClass ItemClass);
```

Description

The Create method takes two parameters: the name of an instance object descended from TPersistent and the name of a TCollectionItem descendant class. The first parameter is the owner of the TOwnedCollection instance. The second parameter determines the class of the items created by the Add method.

See Also

Add

28.1.6.3 TListControlItems Methods

The methods of the TListControlItems class are listed here.

Protected Methods

	Name	Description
	CompareItems (see page 1050)	Returns the relative order of two items in the collection.
	Update (see page 1051)	Updates the collection to reflect changes to its items.

Public Methods

	Name	Description
	Add (see page 1049)	Adds a new item to the collection.
	CustomSort (see page 1050)	Sorts the items in the collection using a specified comparison function.
	Sort (see page 1051)	Sorts the items in the collection.

28.1.6.3.1 TListControlItems.Add Method

Adds a new item to the collection.

Pascal

```
function Add: TListControlItem;
```

C++

```
__fastcall TListControlItem Add();
```

Description

Call Add to add a new item to the collection. Add instantiates an object of the type that was specified when the TListControlItems (see page 1047) instance was created. It returns the newly created object.

See Also

Items (see page 1052)

Delete

28.1.6.3.2 TListControlItems.CompareItems Method

Returns the relative order of two items in the collection.

Pascal

```
function CompareItems(I1: TListControlItem; I2: TListControlItem): Integer; virtual;
```

C++

```
virtual __fastcall int CompareItems(TListControlItem I1, TListControlItem I2);
```

Description

CompareItems is used internally to provide the default sort order for items in the list. It is used when the SortType (see page 1053) property is not stNone. The Sort (see page 1051) method also uses this as a comparison function.

If I1 is greater than I2, CompareItems returns a value greater than 0. If I1 is the same as I2, CompareItems returns 0. If I1 is less than I2, CompareItems returns a value less than 0.

CompareItems always uses an OnCompare (see page 1054) event handler if it is assigned. Otherwise, it compares the captions of the items, taking into account the CaseSensitive (see page 1052) property. Descendant classes can override CompareItems to provide a different default sort order.

See Also

SortType (see page 1053)

CustomSort (see page 1050)

Sort (see page 1051)

CaseSensitive (see page 1052)

OnCompare (see page 1054)

28.1.6.3.3 TListControlItems.CustomSort Method

Sorts the items in the collection using a specified comparison function.

Pascal

```
procedure CustomSort(Compare: TListItemsCompare);
```

C++

```
__fastcall CustomSort(TListItemsCompare Compare);
```

Description

Call Sort (see page 1051) to sort the items in the collection based on the comparison function that is the value of the Compare parameter.

Compare is a function that compares two items in the list. Its List parameter is this TListControlItems (see page 1047) object.

The Index1 and Index2 parameters are the indexes of the items to compare in the Items (see page 1052) property array. This comparison function should return a value less than 0 if the first item is less than the second, a value greater than 0 if it is greater than the second, and 0 if the two items are equivalent.

Note: CustomSort sorts the items in the list, but does not maintain the sort order when new items are added. To provide a custom sort order that persists when new items are added, set the SortType (see page 1053) property to stData and supply an OnCompare (see page 1054) event handler.

See Also

SortType (see page 1053)

Sort (see page 1051)

OnCompare (see page 1054)

Items (see page 1052)

28.1.6.3.4 TListControlItems.Sort Method

Sorts the items in the collection.

Pascal

```
procedure Sort;
```

C++

```
__fastcall Sort();
```

Description

Call Sort to sort the items in the collection. Sort uses the comparison function specified by the protected CompareItems (see page 1050) method to determine the sort order.

Note: Sort sorts the items in the list, but does not maintain the sort order when new items are added. To impose a sort order that persists when new items are added, set the SortType (see page 1053) property to a value other than stNone and, if desired, supply an OnCompare (see page 1054) event handler.

See Also

SortType (see page 1053)

CustomSort (see page 1050)

OnCompare (see page 1054)

Items (see page 1052)

CompareItems (see page 1050)

28.1.6.3.5 TListControlItems.Update Method

Updates the collection to reflect changes to its items.

Pascal

```
procedure Update(Item: TCollectionItem); override;
```

C++

```
virtual __fastcall Update(TCollectionItem Item);
```

Description

Override Update in a descendant class to make any necessary changes when the items in the collection change. This method is called automatically when an update is complete.

Item identifies the item that changed. If the Item parameter is nil (Delphi) or NULL (C++), then the change affects more than one item in the collection.

As implemented in TCollection, Update does nothing. Descendant classes override this method to make any necessary adjustments.

See Also

Changed

Changed

28.1.6.4 TListControlItems Properties

The properties of the TListControlItems class are listed here.

Public Properties

	Name	Description
	Items (see page 1052)	Lists the items in the collection.

Published Properties

	Name	Description
	CaseSensitive (see page 1052)	Controls whether items are sorted in a case-sensitive or case-insensitive manner.
	SortType (see page 1053)	Indicates how the items should be sorted.

28.1.6.4.1 TListControlItems.CaseSensitive Property

Controls whether items are sorted in a case-sensitive or case-insensitive manner.

Pascal

```
property CaseSensitive: Boolean;
```

C++

```
__property Boolean CaseSensitive;
```

Description

Use CaseSensitive to indicate whether the case of item captions should be considered when sorting the items in the list.

See Also

Items (see page 1052)

SortType (see page 1053)

Caption (see page 1046)

28.1.6.4.2 TListControlItems.Items Property

Lists the items in the collection.

Pascal

```
property Items [const Index: Integer]: TListControlItem;
```

C++

```
__property TListControlItem Items[int const Index];
```

Description

Use Items to access the items in this collection.

Index indicates the item to access, where 0 is the index of the first item, 1 is the index of the second item, and so on. The Count property indicates the maximum number of items.

Note: In Delphi code, Items is the default property for TListControlItems (see page 1047). This means that the property name can be omitted when using the Items property. For example,

```
StaticListAction.Items.Items[0].Caption := Edit1.Text;
```

Note: can also be written

```
StaticListAction.Items[0].Caption := Edit1.Text;
```

Note: Items is a read-only property. To add or delete items from the collection, use the Add (see page 1049) and Delete methods.

See Also

Add (see page 1049)

Delete

Count

28.1.6.4.3 TListControlItems.SortType Property

Indicates how the items should be sorted.

Pascal

```
property SortType: TListItemsSortType;
```

C++

```
__property TListItemsSortType SortType;
```

Description

Use SortType to indicate how you want the items sorted, if at all. The following table lists the possible values:

Value	Meaning
stNone	Items (see page 1052) are not automatically sorted. You can still sort the items explicitly by calling the Sort (see page 1051) or CustomSort (see page 1050) method.
stData	Items (see page 1052) are sorted based on the value of their Data property. You must supply an OnCompare (see page 1054) event handler to specify the sort order.
stText	Items (see page 1052) are sorted base on the value of their Caption property. The CaseSensitive (see page 1052) property specifies whether comparisons are case-sensitive.
stBoth	Items (see page 1052) are sorted based on an OnCompare (see page 1054) event handler if one is assigned, otherwise based on the value of their Caption property.

See Also

Items (see page 1052)

OnCompare (see page 1054)

Data (see page 1047)

CaseSensitive (see page 1052)

CustomSort (see page 1050)

Sort (see page 1051)

28.1.6.5 TListControlItems Events

The events of the TListControlItems class are listed here.

Published Events

	Name	Description
	OnCompare (see page 1054)	Occurs to when two items need to be compared for sorting the collection.

28.1.6.5.1 TListControlItems.OnCompare Event

Occurs to when two items need to be compared for sorting the collection.

Pascal

```
property OnCompare: TListCompareEvent;
```

C++

```
__property TListCompareEvent OnCompare;
```

Description

Write an OnCompare event handler to indicate the sort order when the SortType (see page 1053) property is stData or stBoth.

The OnCompare event handler compares the items passed as the Item1 and Item2 parameters. If Item1 is the same as Item2 in the sort order, the event handler should return 0. If Item1 is less than Item2, the event handler should return a value less than 0. If Item1 is greater than Item2, the event handler should return value greater than 0.

See Also

SortType (see page 1053)

28.1.7 TStaticListAction Class

TStaticListAction provides items to client controls.

Class Hierarchy

Pascal

```
TStaticListAction = class(TCustomStaticListAction);
```

C++

```
class TStaticListAction : public TCustomStaticListAction;
```

File

ListActns

Description

TStaticListAction supplies items to client controls such as list views, list boxes, or combo boxes. TStaticListAction maintains a list of items in its Items (see page 1059) property. This lets you specify, at design time, the items it supplies to clients.

When an application sets the Active (see page 1056) property of a list action to true, the action fills all client controls with items. Each item may include text, an image, and a pointer to data. Each client control determines which of these features of the item it accepts.

When the user selects an item in one of the client controls, the client "executes" the list action, which causes it to generate an OnItemSelected (see page 1062) event.

When a TStaticListAction is added to an action manager, it uses the action to generate a drop-down list on a tool bar.

See Also

TVirtualListAction (see page 1066)

TCustomCombo

TCustomListView

TCustomListBox

TCustomComboBox

28.1.7.1 TStaticListAction Members

The following tables list the members exposed by TStaticListAction.

Protected Events

	Name	Description
	OnGetItemCount (see page 1027)	Occurs when the list action needs to determine the number of items it supplies to client controls.

Protected Methods

	Name	Description
V	GetCount (see page 1021)	Returns the value of the Count (see page 1024) property.
V	GetCount (see page 1030)	Returns the value of the Count property.
	GetItem (see page 1030)	Fills in a specified item to match the corresponding item in the Items (see page 1032) list.
V	GetItemClass (see page 1030)	Returns the class type for the items in the Items (see page 1032) property array.
V	GetString (see page 1022)	Returns the value of the Strings (see page 1026) property.
V	GetString (see page 1031)	Returns the value of the Strings property.
V	Loaded (see page 1023)	Initializes the action after it is loaded into memory.
V	SetString (see page 1023)	Sets the value of the Strings (see page 1026) property.
V	SetString (see page 1031)	Sets the value of the Strings property.

Protected Properties

	Name	Description
	Clients (see page 1032)	This is Clients, a member of class TCustomStaticListAction.
R	Loading (see page 1026)	Indicates whether the action is currently adding items to client controls.

Public Methods

	Name	Description
V	Create (see page 1020)	Creates an instance of TCustomListAction (see page 1019).
V	Create (see page 1029)	Creates an instance of TCustomStaticListAction (see page 1027).

	Destroy (see page 1029)	Frees the memory associated with the TCustomStaticListAction (see page 1027) object.
	ExecuteTarget (see page 1021)	Generates an OnItemSelected (see page 1027) event.
	HandlesTarget (see page 1022)	Indicates whether the action can execute its function.

Public Properties

	Name	Description
	Count (see page 1024)	Indicates the number of items this list action represents.
	Count (see page 1032)	Indicates the number of items this list action represents.
	Strings (see page 1026)	Lists the Strings that are the values of the items this action supplies.

Published Events

	Name	Description
	OnGetItem (see page 1061)	Occurs when the static list action supplies an item to its client controls.
	OnHint (see page 1061)	Occurs when the mouse pauses over a client control or menu item.
	OnItemSelected (see page 1062)	Occurs when a client control executes the action because a user selected one of the items.
	OnUpdate (see page 1062)	Occurs when the application is idle or when the action list updates.

Published Properties

	Name	Description
	Active (see page 1056)	Indicates whether the list action has supplied items to its client controls.
	Caption (see page 1057)	Represents the caption of client controls and menu items.
	Enabled (see page 1057)	Indicates whether client controls and menu items are enabled.
	HelpContext (see page 1058)	Indicates the help context ID for client controls and menu items.
	Hint (see page 1058)	Indicates the Help hint for client controls and menu items.
	Images (see page 1058)	Lists the images that the list action supplies along with items.
	ItemIndex (see page 1059)	Specifies the index of the selected item.
	Items (see page 1059)	Lists the items that this list action supplies to clients.
	SecondaryShortCuts (see page 1060)	Specifies the short cuts (in addition to ShortCut) for triggering clients.
	ShortCut (see page 1060)	Specifies the ShortCut property for client menu items.
	Visible (see page 1060)	Specifies the Visible property for client controls and menu items.

28.1.7.2 TStaticListAction Properties

The properties of the TStaticListAction class are listed here.

Published Properties

	Name	Description
	Active (see page 1056)	Indicates whether the list action has supplied items to its client controls.
	Caption (see page 1057)	Represents the caption of client controls and menu items.
	Enabled (see page 1057)	Indicates whether client controls and menu items are enabled.
	HelpContext (see page 1058)	Indicates the help context ID for client controls and menu items.
	Hint (see page 1058)	Indicates the Help hint for client controls and menu items.
	Images (see page 1058)	Lists the images that the list action supplies along with items.
	ItemIndex (see page 1059)	Specifies the index of the selected item.
	Items (see page 1059)	Lists the items that this list action supplies to clients.
	SecondaryShortCuts (see page 1060)	Specifies the short cuts (in addition to ShortCut) for triggering clients.
	ShortCut (see page 1060)	Specifies the ShortCut property for client menu items.
	Visible (see page 1060)	Specifies the Visible property for client controls and menu items.

28.1.7.2.1 TStaticListAction.Active Property

Indicates whether the list action has supplied items to its client controls.

Pascal

```
property Active: Boolean;
```

C++

```
__property Boolean Active;
```

Description

Set Active to true to cause the list action to fill all client controls with the items. These Items are supplied by calling the GetItem method, iterating over every index from 0 to Count.

See Also

GetItem (see page 1035)

Items

Items

Count (see page 1024)

Items

Strings (see page 1026)

28.1.7.2.2 TStaticListAction.Caption Property

Represents the caption of client controls and menu items.

Pascal

```
property Caption;
```

C++

```
__property Caption;
```

Description

Caption holds the string that is the caption for the action when it is set. The value of Caption is propagated to client controls and menu items.

See Also

Caption

Caption (see page 1396)

28.1.7.2.3 TStaticListAction.Enabled Property

Indicates whether client controls and menu items are enabled.

Pascal

```
property Enabled;
```

C++

```
__property Enabled;
```

Description

Enabled specifies the enabled state for the action. The value of Enabled is propagated to client controls and menu items.

See Also

Enabled

Enabled (see page 1406)

28.1.7.2.4 TStaticListAction.HelpContext Property

Indicates the help context ID for client controls and menu items.

Pascal

```
property HelpContext;
```

C++

```
__property HelpContext;
```

Description

HelpContext specifies the help context for the action when it is set. This value is propagated to client controls and menu items.

When a client control has focus and the user presses the F1 key, the control uses the value of HelpContext for this associated action to invoke the online help targeting a specific topic. The target topic is uniquely identified by a nonzero HelpContext value. Leave the HelpContext property of the client control at its default value of zero. This allows multiple controls (and multiple types of controls) to be associated with the same help topic through the one action object.

Note: HelpContext is only used when the HelpType property of this action or the client control is htInteger.

See Also

HelpType

HelpKeyword

28.1.7.2.5 TStaticListAction.Hint Property

Indicates the Help hint for client controls and menu items.

Pascal

```
property Hint;
```

C++

```
__property Hint;
```

Description

Hint holds the string that is the hint for that action when it is set. This value is propagated to client controls and menu items.

See Also

Hint

Hint (see page 1410)

28.1.7.2.6 TStaticListAction.Images Property

Lists the images that the list action supplies along with items.

Pascal

```
property Images: TCustomImageList;
```

C++

```
__property TCustomImageList Images;
```

Description

For each item that the list action supplies to clients, it can supply an image index along with other information such as the text value. The supplied image index refers to an image in the Images property.

Note: List actions do not supply images to all clients. The Images property controls the SmallImages property of a list view or the Images property of a TComboBoxEx descendant.

See Also

SmallImages

Images

28.1.7.2.7 TStaticListAction.ItemIndex Property

Specifies the index of the selected item.

Pascal

```
property ItemIndex: Integer;
```

C++

```
__property int ItemIndex;
```

Description

ItemIndex represents the index of the selected item in all client controls that descend from TCustomListControl. Setting ItemIndex causes these clients to change their selection. When the user updates the selection using one of these clients, TCustomListAction automatically updates the others so that they select the same item.

Note: List actions do not supply ItemIndex to all clients. The ItemIndex property controls only the ItemIndex property of clients that descend from TCustomListControl.

See Also

OnItemSelected (see page 1027)

ItemIndex

28.1.7.2.8 TStaticListAction.Items Property

Lists the items that this list action supplies to clients.

Pascal

```
property Items: TStaticListItems;
```

C++

```
__property TStaticListItems Items;
```

Description

Set Items to indicate the items that this action supplies to clients. At design time, you can use the collection editor to add items to the list and set their properties.

See Also

Count (see page 1032)

Strings (see page 1026)

28.1.7.2.9 TStaticListAction.SecondaryShortCuts Property

Specifies the short cuts (in addition to ShortCut) for triggering clients.

Pascal

```
property SecondaryShortCuts;
```

C++

```
__property SecondaryShortCuts;
```

Description

Actions can execute in response to multiple short cut key combinations. SecondaryShortCuts lists all the short cut key combinations (other than the one specified by the ShortCut property) that can trigger the action. This lets you specify shortcuts for clients that do not have a ShortCut property or provide additional, secondary shortcuts.

When the user types one of the key combinations listed in SecondaryShortCuts, the action's Execute method is called.

See Also

ShortCut

Execute

28.1.7.2.10 TStaticListAction.ShortCut Property

Specifies the ShortCut property for client menu items.

Pascal

```
property ShortCut;
```

C++

```
__property ShortCut;
```

Description

ShortCut holds the value that is the shortcut for the action when it is set. This value is propagated to client menu items.

See Also

ShortCut (see page 1415)

SecondaryShortCuts

28.1.7.2.11 TStaticListAction.Visible Property

Specifies the Visible property for client controls and menu items.

Pascal

```
property Visible;
```

C++

```
__property Visible;
```

Description

Visible specifies the visible state for the action. This value is propagated to client controls and menu items.

See Also

Visible (see page 1416)

Visible

28.1.7.3 TStaticListAction Events

The events of the TStaticListAction class are listed here.

Published Events

	Name	Description
	OnGetItem (see page 1061)	Occurs when the static list action supplies an item to its client controls.
	OnHint (see page 1061)	Occurs when the mouse pauses over a client control or menu item.
	OnItemSelected (see page 1062)	Occurs when a client control executes the action because a user selected one of the items.
	OnUpdate (see page 1062)	Occurs when the application is idle or when the action list updates.

28.1.7.3.1 TStaticListAction.OnGetItem Event

Occurs when the static list action supplies an item to its client controls.

Pascal

```
property OnGetItem: TGetItemEvent;
```

C++

```
__property TGetItemEvent OnGetItem;
```

Description

Write an OnGetItem event handler to customize the properties of an item before it is supplied to client controls. The GetItem method generates an OnGetItem event for every item it supplies to clients, passing in the current item from the Items property. You can change that item's properties in the event handler before it is added to client controls.

Sender is the list action that is supplying data to client controls.

Index indicates which item is requested, where 0 is the first item, 1 is the second item and so on up to the index that is one less than the value of the Count property.

Item is the item that the action is about to supply to the user. You can change any of its properties before it is passed on to the action's client.

See Also

Count (see page 1032)

Items (see page 1059)

Images (see page 1025)

Objects

28.1.7.3.2 TStaticListAction.OnHint Event

Occurs when the mouse pauses over a client control or menu item.

Pascal

```
property OnHint;
```

C++

```
__property OnHint;
```

Description

Use OnHint to override the default hint that appears when the user pauses the mouse over a client control or menu item. The OnHint event handler can change the hint that appears (by changing the HintStr parameter), or display the hint in some application-specific way.

28.1.7.3.3 TStaticListAction.OnItemSelected Event

Occurs when a client control executes the action because a user selected one of the items.

Pascal

```
property OnItemSelected: TItemSelectedEvent;
```

C++

```
__property TItemSelectedEvent OnItemSelected;
```

Description

Write an OnItemSelected event handler to respond when the user selects one of the items supplied to client controls. Before this event is called, the list action first updates the ItemIndex property of any client controls that descend from TCustomListControl.

Sender is the list action that supplies items to the control in which an item was selected.

Control is the control in which the user selected an item.

See Also

ExecuteTarget (see page 1021)

ItemIndex (see page 1025)

28.1.7.3.4 TStaticListAction.OnUpdate Event

Occurs when the application is idle or when the action list updates.

Pascal

```
property OnUpdate;
```

C++

```
__property OnUpdate;
```

Description

Write an OnUpdate event handler to execute centralized code while an application is idle. For example, actions may want to update enabling and disabling, or checking and unchecking of client targets.

C++ Examples:

```
/*
When the application is idle, the OnUpdate event occurs for
every action that is linked to a visible control or menu
item that is showing. This provides an opportunity for
applications to execute centralized code for enabling and
disabling, checking and unchecking, and so on. For example,
the following code illustrates the OnUpdate event handler
for an action that is "checked" when the toolbar is visible.
This example requires a TActionList (with one TAction) and a
TToolBar on a form. Double click
```

```
on the TActionList and create an TAction named Action1.  The
TAction has an OnUpdate event handler called Action1Update.
Then set an object's (such as a button or menu item) Action
to Action1.
*/
Boolean checked;

void __fastcall TForm1::Action1Update(TObject *Sender)
{
   // Indicate whether ToolBar1 is currently visible.
   TAction* pa = dynamic_cast<TAction*>(Sender);
   if (pa)
   {
     if (checked != ToolBar1->Visible)
     {
       lbOther->Items->Add("Event Action1Update: toolbar visibility changed");
     }
     checked = ToolBar1->Visible;
   }
//   pa->Checked = True;  // set this statically and don't change it!
}
```

Delphi Examples:

```
{
When the application is idle, the OnUpdate event occurs for
every action that is linked to a visible control or menu
item that is showing. This provides an opportunity for
applications to execute centralized code for enabling and
disabling, checking and unchecking, and so on. For example,
the following code illustrates the OnUpdate event handler
for an action that is "checked" when the toolbar is visible.
This example requires a TActionList (with one TAction) and a
TToolBar on a form. This example requires a TActionList
(with one TAction) and a TToolBar on a form.  Double click
on the TActionList and create an TAction named Action1.  The
TAction has an OnUpdate event handler called Action1Update.
Then set an object's (such as a button or menu item) Action
to Action1.
}
procedure TMainForm.Action1Update(Sender: TObject);
begin
  if ((Sender as TAction).Checked <> ToolBar1.Visible) then
    lbOther.Items.Add('Event Action1Update: toolbar visibility changed');
    { Indicate whether ToolBar1 is currently visible }
    (Sender as TAction).Checked := ToolBar1.Visible;
end;
```

See Also

UpdateAction

28.1.8 TStaticListItems Class

TStaticListItems represents the items that TStaticListAction (see page 1054) supplies to client controls.

Class Hierarchy

Pascal

```
TStaticListItems = class(TListControlItems);
```

C++

```
class TStaticListItems : public TListControlItems;
```

File

ListActns

Description

TStaticListItems manages a collection of items for a TStaticListAction (see page 1054) component. Each item in the collection can include a text value, associated image, and a data pointer. If the associated TStaticListAction (see page 1054) is active, then when items are added to or deleted from TStaticListItems, the list action's clients are updated so that they reflect the current set of items.

See Also

TStaticListAction (see page 1054)

28.1.8.1 TStaticListItems Members

The following tables list the members exposed by TStaticListItems.

Protected Methods

	Name	Description
	CompareItems (see page 1050)	Returns the relative order of two items in the collection.
	Notify (see page 1065)	Responds when items are added to or removed from the collection.
	Update (see page 1065)	Updates the collection to reflect changes to its items.

Public Methods

	Name	Description
	Add (see page 1049)	Adds a new item to the collection.
	Create (see page 1049)	Creates and initializes a TOwnedCollection instance.
	CustomSort (see page 1050)	Sorts the items in the collection using a specified comparison function.
	Sort (see page 1051)	Sorts the items in the collection.

Public Properties

	Name	Description
	Items (see page 1052)	Lists the items in the collection.

Published Events

	Name	Description
	OnCompare (see page 1054)	Occurs to when two items need to be compared for sorting the collection.

Published Properties

	Name	Description
	CaseSensitive (see page 1052)	Controls whether items are sorted in a case-sensitive or case-insensitive manner.
	SortType (see page 1053)	Indicates how the items should be sorted.

28.1.8.2 TStaticListItems Methods

The methods of the TStaticListItems class are listed here.

Protected Methods

	Name	Description
	Notify (see page 1065)	Responds when items are added to or removed from the collection.
	Update (see page 1065)	Updates the collection to reflect changes to its items.

28.1.8.2.1 TStaticListItems.Notify Method

Responds when items are added to or removed from the collection.

Pascal

```
procedure Notify(Item: TCollectionItem; Action: TCollectionNotification); override;
```

C++

```
virtual __fastcall Notify(TCollectionItem Item, TCollectionNotification Action);
```

Description

Notify is called automatically when the items in the collection change.

Item is the item that was just added to or that is about to be removed from the collection.

Action indicates whether item was added, is about to be removed, or is about to be deleted:

As implemented in TCollection, Notify calls Added when `Action` is cnAdded and calls Deleting when `Action` is cnDeleting. TCollection ignores the cnExtracting action. Descendant classes can override Notify to modify this behavior.

See Also

Added

Deleting

28.1.8.2.2 TStaticListItems.Update Method

Updates the collection to reflect changes to its items.

Pascal

```
procedure Update(Item: TCollectionItem); override;
```

C++

```
virtual __fastcall Update(TCollectionItem Item);
```

Description

Override Update in a descendant class to make any necessary changes when the items in the collection change. This method is called automatically when an update is complete.

Item identifies the item that changed. If the Item parameter is nil (Delphi) or NULL (C++), then the change affects more than one item in the collection.

As implemented in TCollection, Update does nothing. Descendant classes override this method to make any necessary adjustments.

See Also

Changed

Changed

28.1.9 TVirtualListAction Class

TVirtualListAction provides items to a client controls.

Class Hierarchy

Pascal

```
TVirtualListAction = class(TCustomVirtualListAction);
```

C++

```
class TVirtualListAction : public TCustomVirtualListAction;
```

File

ListActns

Description

TVirtualListAction supplies items dynamically at runtime to client controls such as list views, list boxes, or combo boxes. These items are supplied by the code you supply to an OnGetItem (see page 1072) event handler.

When an application sets the Active (see page 1068) property of a list action to true, the action fills all client controls with items. Each item may include text, an image, and a pointer to data. Each client control determines which of these features of the item it accepts.

When the user selects an item in one of the client controls, the client "executes" the list action, which causes it to generate an OnItemSelected (see page 1073) event.

When a TVirtualListAction is added to an action manager, it uses the action to generate a drop-down list on a tool bar.

See Also

TStaticListAction (see page 1054)

TCustomCombo

TCustomListView

TCustomListBox

TCustomComboBox

28.1.9.1 TVirtualListAction Members

The following tables list the members exposed by TVirtualListAction.

Protected Methods

	Name	Description
	GetCount (see page 1021)	Returns the value of the Count (see page 1024) property.
	GetItem (see page 1035)	Provides information about a specified item.
	GetString (see page 1022)	Returns the value of the Strings (see page 1026) property.
	GetString (see page 1035)	Returns the value of the Strings property.
	Loaded (see page 1023)	Initializes the action after it is loaded into memory.
	SetString (see page 1023)	Sets the value of the Strings (see page 1026) property.

Protected Properties

	Name	Description
	Loading (see page 1026)	Indicates whether the action is currently adding items to client controls.

Public Methods

	Name	Description
	Create (see page 1020)	Creates an instance of TCustomListAction (see page 1019).
	ExecuteTarget (see page 1021)	Generates an OnItemSelected (see page 1027) event.
	HandlesTarget (see page 1022)	Indicates whether the action can execute its function.

Public Properties

	Name	Description
	Count (see page 1024)	Indicates the number of items this list action represents.
	Count (see page 1036)	Indicates the number of items this list action represents.
	Strings (see page 1026)	Lists the Strings that are the values of the items this action supplies.

Published Events

	Name	Description
	OnGetItem (see page 1072)	Occurs when the list action needs to supply an item to its client controls.
	OnGetItemCount (see page 1072)	Occurs when the list action needs to determine the number of items it supplies to client controls.
	OnHint (see page 1073)	Occurs when the mouse pauses over a client control or menu item.
	OnItemSelected (see page 1073)	Occurs when a client control executes the action because a user selected one of the items.

Published Properties

	Name	Description
	Active (see page 1068)	Indicates whether the list action has supplied items to its client controls.
	Caption (see page 1068)	Represents the caption of client controls and menu items.
	Enabled (see page 1068)	Indicates whether client controls and menu items are enabled.
	HelpContext (see page 1069)	Indicates the help context ID for client controls and menu items.
	Hint (see page 1069)	Indicates the Help hint for client controls and menu items.
	Images (see page 1070)	Lists the images that the list action supplies along with items.
	ItemIndex (see page 1070)	Specifies the index of the selected item.
	SecondaryShortCuts (see page 1070)	Specifies the short cuts (in addition to ShortCut) for triggering clients.
	ShortCut (see page 1071)	Specifies the ShortCut property for client menu items.
	Visible (see page 1071)	Specifies the Visible property for client controls and menu items.

28.1.9.2 TVirtualListAction Properties

The properties of the TVirtualListAction class are listed here.

Published Properties

	Name	Description
	Active (see page 1068)	Indicates whether the list action has supplied items to its client controls.
	Caption (see page 1068)	Represents the caption of client controls and menu items.
	Enabled (see page 1068)	Indicates whether client controls and menu items are enabled.
	HelpContext (see page 1069)	Indicates the help context ID for client controls and menu items.
	Hint (see page 1069)	Indicates the Help hint for client controls and menu items.
	Images (see page 1070)	Lists the images that the list action supplies along with items.
	ItemIndex (see page 1070)	Specifies the index of the selected item.
	SecondaryShortCuts (see page 1070)	Specifies the short cuts (in addition to ShortCut) for triggering clients.
	ShortCut (see page 1071)	Specifies the ShortCut property for client menu items.

	Visible (see page 1071)	Specifies the Visible property for client controls and menu items.

28.1.9.2.1 TVirtualListAction.Active Property

Indicates whether the list action has supplied items to its client controls.

Pascal

```
property Active: Boolean;
```

C++

```
__property Boolean Active;
```

Description

Set Active to true to cause the list action to fill all client controls with the items. These Items are supplied by calling the GetItem method, iterating over every index from 0 to Count.

See Also

GetItem (see page 1035)

Items

Items

Count (see page 1024)

Items

Strings (see page 1026)

28.1.9.2.2 TVirtualListAction.Caption Property

Represents the caption of client controls and menu items.

Pascal

```
property Caption;
```

C++

```
__property Caption;
```

Description

Caption holds the string that is the caption for the action when it is set. The value of Caption is propagated to client controls and menu items.

See Also

Caption

Caption (see page 1396)

28.1.9.2.3 TVirtualListAction.Enabled Property

Indicates whether client controls and menu items are enabled.

Pascal

```
property Enabled;
```

C++

```
__property Enabled;
```

Description

Enabled specifies the enabled state for the action. The value of Enabled is propagated to client controls and menu items.

See Also

Enabled

Enabled (see page 1406)

28.1.9.2.4 TVirtualListAction.HelpContext Property

Indicates the help context ID for client controls and menu items.

Pascal

```
property HelpContext;
```

C++

```
__property HelpContext;
```

Description

HelpContext specifies the help context for the action when it is set. This value is propagated to client controls and menu items.

When a client control has focus and the user presses the F1 key, the control uses the value of HelpContext for this associated action to invoke the online help targeting a specific topic. The target topic is uniquely identified by a nonzero HelpContext value. Leave the HelpContext property of the client control at its default value of zero. This allows multiple controls (and multiple types of controls) to be associated with the same help topic through the one action object.

Note: HelpContext is only used when the HelpType property of this action or the client control is htInteger.

See Also

HelpType

HelpKeyword

28.1.9.2.5 TVirtualListAction.Hint Property

Indicates the Help hint for client controls and menu items.

Pascal

```
property Hint;
```

C++

```
__property Hint;
```

Description

Hint holds the string that is the hint for that action when it is set. This value is propagated to client controls and menu items.

See Also

Hint

Hint (see page 1410)

28.1.9.2.6 TVirtualListAction.Images Property

Lists the images that the list action supplies along with items.

Pascal

```
property Images: TCustomImageList;
```

C++

```
__property TCustomImageList Images;
```

Description

For each item that the list action supplies to clients, it can supply an image index along with other information such as the text value. The supplied image index refers to an image in the Images property.

Note: List actions do not supply images to all clients. The Images property controls the SmallImages property of a list view or the Images property of a TComboBoxEx descendant.

See Also

SmallImages

Images

28.1.9.2.7 TVirtualListAction.ItemIndex Property

Specifies the index of the selected item.

Pascal

```
property ItemIndex: Integer;
```

C++

```
__property int ItemIndex;
```

Description

ItemIndex represents the index of the selected item in all client controls that descend from TCustomListControl. Setting ItemIndex causes these clients to change their selection. When the user updates the selection using one of these clients, TCustomListAction automatically updates the others so that they select the same item.

Note: List actions do not supply ItemIndex to all clients. The ItemIndex property controls only the ItemIndex property of clients that descend from TCustomListControl.

See Also

OnItemSelected (see page 1027)

ItemIndex

28.1.9.2.8 TVirtualListAction.SecondaryShortCuts Property

Specifies the short cuts (in addition to ShortCut) for triggering clients.

Pascal

```
property SecondaryShortCuts;
```

C++

```
__property SecondaryShortCuts;
```

Description

Actions can execute in response to multiple short cut key combinations. SecondaryShortCuts lists all the short cut key combinations (other than the one specified by the ShortCut property) that can trigger the action. This lets you specify shortcuts for clients that do not have a ShortCut property or provide additional, secondary shortcuts.

When the user types one of the key combinations listed in SecondaryShortCuts, the action's Execute method is called.

See Also

ShortCut

Execute

28.1.9.2.9 TVirtualListAction.ShortCut Property

Specifies the ShortCut property for client menu items.

Pascal

```
property ShortCut;
```

C++

```
__property ShortCut;
```

Description

ShortCut holds the value that is the shortcut for the action when it is set. This value is propagated to client menu items.

See Also

ShortCut (see page 1415)

SecondaryShortCuts

28.1.9.2.10 TVirtualListAction.Visible Property

Specifies the Visible property for client controls and menu items.

Pascal

```
property Visible;
```

C++

```
__property Visible;
```

Description

Visible specifies the visible state for the action. This value is propagated to client controls and menu items.

See Also

Visible (see page 1416)

Visible

28.1.9.3 TVirtualListAction Events

The events of the TVirtualListAction class are listed here.

Published Events

	Name	Description
	OnGetItem (see page 1072)	Occurs when the list action needs to supply an item to its client controls.
	OnGetItemCount (see page 1072)	Occurs when the list action needs to determine the number of items it supplies to client controls.
	OnHint (see page 1073)	Occurs when the mouse pauses over a client control or menu item.
	OnItemSelected (see page 1073)	Occurs when a client control executes the action because a user selected one of the items.

28.1.9.3.1 TVirtualListAction.OnGetItem Event

Occurs when the list action needs to supply an item to its client controls.

Pascal

```
property OnGetItem: TGetVirtualItemEvent;
```

C++

```
__property TGetVirtualItemEvent OnGetItem;
```

Description

Write an OnGetItem event handler to provide the items that the virtual list action supplies to clients. When you set the Active property to True, the virtual list action generates an OnGetItem event for every index from 0 to Count – 1. In the event handler, supply the string value, image, and data that corresponds to the index specified by Index.

Sender is the list action that is supplying data to client controls.

Index indicates which item is requested, where 0 is the first item, 1 is the second item and so on up to the index that is one less than the value of the Count property.

Value returns the string value of the item. This is the text that appears in the client combo box or list box.

ImageIndex is the index of an image associated with the item. If the item is not associated with an image, ImageIndex should return –1. Otherwise, it should return the index of the desired image in the Images property.

Data returns a pointer to application-defined data that is associated with the item. For client controls that store items in a TStrings object, Data is the value of the Objects property for the item.

See Also

Count (see page 1024)

Images (see page 1025)

Objects

OnGetItemCount (see page 1027)

Active (see page 1024)

28.1.9.3.2 TVirtualListAction.OnGetItemCount Event

Occurs when the list action needs to determine the number of items it supplies to client controls.

Pascal

```
property OnGetItemCount: TGetItemCountEvent;
```

C++

```
__property TGetItemCountEvent OnGetItemCount;
```

Description

The GetCount method generates an OnGetItemCount event to supply the value of the Count property. Write an OnGetItemCount event handler to indicate the number of items for which you are supplying values in the OnGetItem event handler.

Sender is the list action that is supplying data to client controls.

Count returns the number of items this list action supplies to client controls.

See Also

GetCount (see page 1021)

Count (see page 1024)

28.1.9.3.3 TVirtualListAction.OnHint Event

Occurs when the mouse pauses over a client control or menu item.

Pascal

```
property OnHint;
```

C++

```
__property OnHint;
```

Description

Use OnHint to override the default hint that appears when the user pauses the mouse over a client control or menu item. The OnHint event handler can change the hint that appears (by changing the HintStr parameter), or display the hint in some application-specific way.

28.1.9.3.4 TVirtualListAction.OnItemSelected Event

Occurs when a client control executes the action because a user selected one of the items.

Pascal

```
property OnItemSelected: TItemSelectedEvent;
```

C++

```
__property TItemSelectedEvent OnItemSelected;
```

Description

Write an OnItemSelected event handler to respond when the user selects one of the items supplied to client controls. Before this event is called, the list action first updates the ItemIndex property of any client controls that descend from TCustomListControl.

Sender is the list action that supplies items to the control in which an item was selected.

Control is the control in which the user selected an item.

See Also

ExecuteTarget (see page 1021)

ItemIndex (see page 1025)

28.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TListItemsSortType (see page 1074)	This is record ListActns.TListItemsSortType.

28.2.1 ListActns.TListItemsSortType Enumeration

Pascal

```
TListItemsSortType = (
  stNone,
  stData,
  stText,
  stBoth
);
```

C++

```
enum TListItemsSortType {
  stNone,
  stData,
  stText,
  stBoth
};
```

File

ListActns

Description

This is record ListActns.TListItemsSortType.

28.3 Types

The following table lists types in this documentation.

Types

Name	Description
TGetItemCountEvent (see page 1075)	This is type ListActns.TGetItemCountEvent.
TGetItemEvent (see page 1075)	This is type ListActns.TGetItemEvent.
TGetVirtualItemEvent (see page 1076)	This is type ListActns.TGetVirtualItemEvent.
TItemSelectedEvent (see page 1076)	This is type ListActns.TItemSelectedEvent.
TListCompareEvent (see page 1076)	This is type ListActns.TListCompareEvent.
TListControlItemClass (see page 1077)	This is type ListActns.TListControlItemClass.
TListItemsCompare (see page 1077)	This is type ListActns.TListItemsCompare.

28.3.1 ListActns.TGetItemCountEvent Type

Pascal

```
TGetItemCountEvent = procedure (Sender: TCustomListAction; var Count: Integer) of object;
```

C++

```
(Sender: TCustomListAction; var Count: Integer) ( TGetItemCountEvent)();
```

File

ListActns

Description

This is type ListActns.TGetItemCountEvent.

28.3.2 ListActns.TGetItemEvent Type

Pascal

```
TGetItemEvent = procedure (Sender: TCustomListAction; const Index: Integer; var Item:
TListControlItem) of object;
```

C++

```
(Sender: TCustomListAction; const Index: Integer; var Item: TListControlItem) (
TGetItemEvent)();
```

File

ListActns

Description

This is type ListActns.TGetItemEvent.

28.3.3 ListActns.TGetVirtualItemEvent Type

Pascal

```
TGetVirtualItemEvent = procedure (Sender: TCustomListAction; const Index: Integer; var Value:
String; var ImageIndex: Integer; var Data: TCustomData) of object;
```

C++

```
(Sender: TCustomListAction; const Index: Integer; var Value: String; var ImageIndex: Integer;
var Data: TCustomData) ( TGetVirtualItemEvent)();
```

File

ListActns

Description

This is type ListActns.TGetVirtualItemEvent.

28.3.4 ListActns.TItemSelectedEvent Type

Pascal

```
TItemSelectedEvent = procedure (Sender: TCustomListAction; Control: TControl) of object;
```

C++

```
(Sender: TCustomListAction; Control: TControl) ( TItemSelectedEvent)();
```

File

ListActns

Description

This is type ListActns.TItemSelectedEvent.

28.3.5 ListActns.TListCompareEvent Type

Pascal

```
TListCompareEvent = function (List: TListControlItems; Item1, Item2: TListControlItem):
Integer of object;
```

C++

```
(List: TListControlItems; Item1, Item2: TListControlItem): Integer ( TListCompareEvent)();
```

File

ListActns

Description

This is type ListActns.TListCompareEvent.

28.3.6 ListActns.TListControlItemClass Type

Pascal

```
TListControlItemClass = class of TListControlItem;
```

C++

```
TListControlItem TListControlItemClass;
```

File

ListActns

Description

This is type ListActns.TListControlItemClass.

28.3.7 ListActns.TListItemsCompare Type

Pascal

```
TListItemsCompare = function (List: TListControlItems; Index1, Index2: Integer): Integer;
```

C++

```
(List: TListControlItems; Index1, Index2: Integer): Integer ( TListItemsCompare)();
```

File

ListActns

Description

This is type ListActns.TListItemsCompare.

29 Mask Namespace

This is namespace Mask.

Topics

Name	Description
Classes (see page 1080)	The following table lists classes in this documentation.
Types (see page 1175)	The following table lists types in this documentation.

Classes

	Name	Description
	EDBEditError (see page 1080)	EDBEditError is the exception class for data that are incompatible with a specified mask.
	TCustomMaskEdit (see page 1080)	TCustomMaskEdit is the base type for edit controls that provide a mask for checking the validity of text typed into the control.
	TMaskEdit (see page 1100)	TMaskEdit implements a generic masked edit control.

Types

Name	Description
TMaskedState (see page 1175)	TMaskedState is a set of flags that describe the current state of a masked edit control.

29.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	EDBEditError (see page 1080)	EDBEditError is the exception class for data that are incompatible with a specified mask.
	TCustomMaskEdit (see page 1080)	TCustomMaskEdit is the base type for edit controls that provide a mask for checking the validity of text typed into the control.
	TMaskEdit (see page 1100)	TMaskEdit implements a generic masked edit control.

29.1.1 EDBEditError Class

EDBEditError is the exception class for data that are incompatible with a specified mask.

Class Hierarchy

Pascal

```
EDBEditError = class(Exception);
```

C++

```
class EDBEditError : public Exception;
```

File

Mask

Description

EDBEditError is raised when an application attempts to use data that are incompatible with the mask specified for a field.

A mask restricts the data that can be put into an edit box or data field. Use the EditMask property to specify a mask.

See Also

TCustomMaskEdit (see page 1080)

TEditMask (see page 1193)

29.1.2 TCustomMaskEdit Class

TCustomMaskEdit is the base type for edit controls that provide a mask for checking the validity of text typed into the control.

Class Hierarchy

Pascal

```
TCustomMaskEdit = class(TCustomEdit);
```

C++

```
class TCustomMaskEdit : public TCustomEdit;
```

File

Mask

Description

TCustomMaskEdit introduces new properties and methods to expand the capabilities of TCustomEdit for validity checking using an edit mask. To create an edit control that takes advantage of the masking capabilities of TCustomMaskEdit, use a TMaskEdit (see page 1100) object. To create an edit control that does not validate the input, use the TEdit object instead.

Do not create instances of TCustomMaskEdit. Use TCustomMaskEdit as a base class when declaring control objects that implement a masked edit control. Properties and methods of TCustomMaskEdit provide basic behavior that descendant classes inherit as well as behavior that components can override to customize their behavior.

See Also

EDBEditError (see page 1080)

TDBEdit

TEdit

TMaskEdit (see page 1100)

29.1.2.1 TCustomMaskEdit Members

The following tables list the members exposed by TCustomMaskEdit.

Protected Methods

	Name	Description
	CheckCursor (see page 1083)	Moves the cursor to the closest editable character and selects it if nothing is currently selected.
V	EditCanModify (see page 1084)	Permits the user to edit control contents.
	GetFirstEditChar (see page 1084)	Returns the index in the EditMask (see page 1092) of the first nonliteral character.
	GetLastEditChar (see page 1085)	Returns the index in the EditMask (see page 1092) of the last nonliteral character.
	GetMaxChars (see page 1085)	Returns the length of the EditText (see page 1095).
	GetNextEditChar (see page 1086)	Returns the first index of a nonliteral character equal to or greater than an offset.
	GetPriorEditChar (see page 1086)	Returns the first index of a nonliteral character equal to or less than an offset.
	GetSel (see page 1087)	Returns the endpoints of the current selection in the EditText (see page 1095).
V	KeyDown (see page 1087)	Supplements the inherited KeyDown to process navigation keys and the Delete key.
V	KeyPress (see page 1088)	Supplements the inherited method to process the Esc and Enter keys.
V	KeyUp (see page 1088)	Key up event dispatcher.
	ReformatText (see page 1089)	Applies the mask NewMask to the underlying text, and updates the internal representation of the mask.
V	Reset (see page 1089)	Returns the EditText (see page 1095) to its value at the time when the edit control last lost focus.
	SetCursor (see page 1090)	Moves the cursor to the first editable character at or before Pos, and selects that character.
	SetSel (see page 1090)	Selects character range.
V	Validate (see page 1091)	Validates a specified string against the current mask.

	ValidateError (see page 1091)	Beeps and raises an EDBEditError (see page 1080) exception.

Protected Properties

	Name	Description
	EditMask (see page 1092)	Specifies the mask that represents what text is valid for the masked edit control.
	MaskState (see page 1096)	Indicates state information about how the mask is being applied.
	MaxLength (see page 1096)	Specifies the maximum number of characters that can appear in the edit control.

Public Methods

	Name	Description
	Clear (see page 1083)	Sets the Text (see page 1097) property to an empty string.
	Create (see page 1082)	Creates an instance of TCustomMaskEdit (see page 1080).
	GetTextLen (see page 1087)	Returns the length of the underlying text.
	ValidateEdit (see page 1091)	Validates the EditText (see page 1095) against the current mask.

Public Properties

	Name	Description
	EditText (see page 1095)	Specifies the text that appears in the window of the masked edit control.
	IsMasked (see page 1096)	Indicates whether a mask exists for the edit control.
	Text (see page 1097)	Denotes the underlying text for the masked edit control before the mask has been applied.

29.1.2.2 TCustomMaskEdit.Create Constructor

Creates an instance of TCustomMaskEdit (see page 1080).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomMaskEdit(TComponent * AOwner);
```

Description

Call Create to instantiate a masked edit control at runtime. AOwner is the component that is responsible for freeing the masked edit control (typically the form). It becomes the value of the Owner property.

See Also

Owner

29.1.2.3 TCustomMaskEdit Methods

The methods of the TCustomMaskEdit class are listed here.

Protected Methods

	Name	Description
	CheckCursor (see page 1083)	Moves the cursor to the closest editable character and selects it if nothing is currently selected.
	EditCanModify (see page 1084)	Permits the user to edit control contents.
	GetFirstEditChar (see page 1084)	Returns the index in the EditMask (see page 1092) of the first nonliteral character.
	GetLastEditChar (see page 1085)	Returns the index in the EditMask (see page 1092) of the last nonliteral character.
	GetMaxChars (see page 1085)	Returns the length of the EditText (see page 1095).
	GetNextEditChar (see page 1086)	Returns the first index of a nonliteral character equal to or greater than an offset.

	GetPriorEditChar (see page 1086)	Returns the first index of a nonliteral character equal to or less than an offset.
	GetSel (see page 1087)	Returns the endpoints of the current selection in the EditText (see page 1095).
V	KeyDown (see page 1087)	Supplements the inherited KeyDown to process navigation keys and the Delete key.
V	KeyPress (see page 1088)	Supplements the inherited method to process the Esc and Enter keys.
V	KeyUp (see page 1088)	Key up event dispatcher.
	ReformatText (see page 1089)	Applies the mask NewMask to the underlying text, and updates the internal representation of the mask.
V	Reset (see page 1089)	Returns the EditText (see page 1095) to its value at the time when the edit control last lost focus.
	SetCursor (see page 1090)	Moves the cursor to the first editable character at or before Pos, and selects that character.
	SetSel (see page 1090)	Selects character range.
V	Validate (see page 1091)	Validates a specified string against the current mask.
V	ValidateError (see page 1091)	Beeps and raises an EDBEditError (see page 1080) exception.

Public Methods

	Name	Description
V	Clear (see page 1083)	Sets the Text (see page 1097) property to an empty string.
	GetTextLen (see page 1087)	Returns the length of the underlying text.
V	ValidateEdit (see page 1091)	Validates the EditText (see page 1095) against the current mask.

29.1.2.3.1 TCustomMaskEdit.CheckCursor Method

Moves the cursor to the closest editable character and selects it if nothing is currently selected.

Pascal

```
procedure CheckCursor;
```

C++

```
__fastcall CheckCursor();
```

Description

CheckCursor is called whenever the position of the cursor may have changed. It ensures that at least one character is always selected. Because editing a masked edit control consists of replacing the blank characters in the mask, there must always be a selection to replace. The selection corresponds to one or more nonliteral characters in the mask, which are either blank or contain characters for the user to reenter.

See Also

GetSel (see page 1087)

SetCursor (see page 1090)

SetSel (see page 1090)

29.1.2.3.2 TCustomMaskEdit.Clear Method

Sets the Text (see page 1097) property to an empty string.

Pascal

```
procedure Clear; override;
```

C++

```
virtual __fastcall Clear();
```

Description

Use Clear to remove all text that was entered into the masked edit control. Clear sets the value of Text (see page 1097) to an empty string. It does not clear the EditMask (see page 1092). If the edit control has a mask, EditText (see page 1095) is not empty, but has all non-literal characters replaced by blanks.

See Also

EditMask (see page 1092)

EditText (see page 1095)

IsMasked (see page 1096)

Reset (see page 1089)

Text (see page 1097)

29.1.2.3.3 TCustomMaskEdit.EditCanModify Method

Permits the user to edit control contents.

Pascal

```
function EditCanModify: Boolean; virtual;
```

C++

```
virtual __fastcall Boolean EditCanModify();
```

Description

The EditCanModify method defined by TCustomMaskEdit (see page 1080) always returns true, allowing the Text (see page 1097) to be edited. EditCanModify exists primarily to provide an interface for a method that can conditionally disallow changes to the Text (see page 1097) by returning false.

See Also

Text (see page 1097)

29.1.2.3.4 TCustomMaskEdit.GetFirstEditChar Method

Returns the index in the EditMask (see page 1092) of the first nonliteral character.

Pascal

```
function GetFirstEditChar: Integer;
```

C++

```
__fastcall int GetFirstEditChar();
```

Description

Use GetFirstEditChar to find the first position of the EditMask (see page 1092) where the user can modify the EditText (see page 1095). 0 specifies the first character, 1 the second character, and so on. If there is no mask, GetFirstEditChar returns zero.

Use GetFirstEditChar as a starting point for programmatically processing all the nonliteral characters in the EditText (see page 1095). Use GetNextEditChar (see page 1086) to continue to loop through the editable characters in the EditText (see page 1095), and GetLastEditChar (see page 1085) to determine when all nonliteral characters have been processed.

See Also

EditMask (see page 1092)

EditText (see page 1095)

GetLastEditChar (see page 1085)

GetNextEditChar (see page 1086)

GetPriorEditChar (see page 1086)

29.1.2.3.5 TCustomMaskEdit.GetLastEditChar Method

Returns the index in the EditMask (see page 1092) of the last nonliteral character.

Pascal

```
function GetLastEditChar: Integer;
```

C++

```
__fastcall int GetLastEditChar();
```

Description

Use GetLastEditChar to find the last position of the EditMask (see page 1092) where the user can modify the EditText (see page 1095). 0 specifies the first character, 1 the second character, and so on. If there is no mask, GetLastEditChar returns the length of the text.

See Also

EditMask (see page 1092)

EditText (see page 1095)

GetFirstEditChar (see page 1084)

GetMaxChars (see page 1085)

GetNextEditChar (see page 1086)

GetPriorEditChar (see page 1086)

29.1.2.3.6 TCustomMaskEdit.GetMaxChars Method

Returns the length of the EditText (see page 1095).

Pascal

```
function GetMaxChars: Integer;
```

C++

```
__fastcall int GetMaxChars();
```

Description

If there is a mask, GetMaxChars returns the length of the EditText (see page 1095), including all blanks. This is the same as MaxLength (see page 1096), because editing in a masked edit control consists of replacing nonliteral characters. If there is no mask, GetMaxChars returns the length of the text in the window. In this case, the value returned by GetMaxChars may be less than MaxLength (see page 1096), which is the maximum number of characters that can be entered into the edit control.

See Also

EditText (see page 1095)

GetLastEditChar (see page 1085)

MaxLength (see page 1096)

Text (see page 1097)

29.1.2.3.7 TCustomMaskEdit.GetNextEditChar Method

Returns the first index of a nonliteral character equal to or greater than an offset.

Pascal

```
function GetNextEditChar(Offset: Integer): Integer;
```

C++

```
__fastcall int GetNextEditChar(int Offset);
```

Description

Call GetNextEditChar to find the first position of the EditMask (see page 1092) at or following Offset where the user can modify the EditText (see page 1095). 0 specifies the first character, 1 the second character, and so on. If there is no nonliteral character at or following Offset, GetNextEditChar returns MaxLength (see page 1096).

Use GetNextEditChar to loop through the editable characters in the EditText (see page 1095), starting with the value returned by GetFirstEditChar (see page 1084) and using GetLastEditChar (see page 1085) to determine when all nonliteral characters have been processed.

See Also

EditMask (see page 1092)

EditText (see page 1095)

GetFirstEditChar (see page 1084)

GetLastEditChar (see page 1085)

GetPriorEditChar (see page 1086)

29.1.2.3.8 TCustomMaskEdit.GetPriorEditChar Method

Returns the first index of a nonliteral character equal to or less than an offset.

Pascal

```
function GetPriorEditChar(Offset: Integer): Integer;
```

C++

```
__fastcall int GetPriorEditChar(int Offset);
```

Description

Call GetPriorEditChar to find the first position of the EditMask (see page 1092) at or preceding Offset, where the user can modify the EditText (see page 1095). 0 specifies the first character, 1 the second character, and so on. If there is no nonliteral character at or preceding Offset, GetPriorEditChar returns the index of the first editable character following Offset.

See Also

EditMask (see page 1092)

EditText (see page 1095)

GetFirstEditChar (see page 1084)

GetLastEditChar (see page 1085)

GetNextEditChar (see page 1086)

29.1.2.3.9 TCustomMaskEdit.GetSel Method

Returns the endpoints of the current selection in the EditText (see page 1095).

Pascal

```
procedure GetSel(var SelStart: Integer; var SelStop: Integer);
```

C++

```
__fastcall GetSel(int SelStart, int SelStop);
```

Description

Use GetSel to obtain the indexes of the first and last selected characters of the EditText (see page 1095), where 0 is the index of the first character. Not all the characters in the selection will necessarily be editable, if the mask contains literal characters. Do not use the resulting indexes on the Text (see page 1097) property if there is a mask, because the EditText (see page 1095) may contain blanks or nonliteral characters not found in the Text (see page 1097).

To programmatically change the selected text, use GetNextEditChar (see page 1086) to cycle through the editable characters of the selection from SelStart to SelStop.

See Also

EditText (see page 1095)

GetNextEditChar (see page 1086)

SelText

SetSel (see page 1090)

Text (see page 1097)

29.1.2.3.10 TCustomMaskEdit.GetTextLen Method

Returns the length of the underlying text.

Pascal

```
function GetTextLen: Integer; reintroduce;
```

C++

```
__fastcall int GetTextLen();
```

Description

Use GetTextLen to determine the length of the underlying text for the masked edit control. The underlying text has not been formatted for display by the mask, and may differ from the EditText (see page 1095) that appears in the window of the control.

See Also

EditText (see page 1095)

GetTextBuf

MaxLength (see page 1096)

Text (see page 1097)

29.1.2.3.11 TCustomMaskEdit.KeyDown Method

Supplements the inherited KeyDown to process navigation keys and the Delete key.

Pascal

```
procedure KeyDown(var Key: Word; Shift: TShiftState); override;
```

C++

```
virtual __fastcall KeyDown(Word Key, TShiftState Shift);
```

Description

After calling the inherited KeyDown, KeyDown ensures that moving the cursor within the masked edit control places the cursor on an editable character, and that deleting the current selection replaces nonliteral characters with blanks. Derived objects can override this method to further supplement the processing that takes place when the user presses a key down.

See Also

CheckCursor (see page 1083)

KeyPress (see page 1088)

KeyUp (see page 1088)

SetSel (see page 1090)

29.1.2.3.12 TCustomMaskEdit.KeyPress Method

Supplements the inherited method to process the Esc and Enter keys.

Pascal

```
procedure KeyPress(var Key: Char); override;
```

C++

```
virtual __fastcall KeyPress(Char Key);
```

Description

After calling the inherited method, KeyPress calls the Reset (see page 1089) method in response to the Esc key, or calls the ValidateEdit (see page 1091) method in response to the Enter key. Derived classes can override this method to further supplement the processing of specific keystrokes.

See Also

KeyDown (see page 1087)

Reset (see page 1089)

ValidateEdit (see page 1091)

29.1.2.3.13 TCustomMaskEdit.KeyUp Method

Key up event dispatcher.

Pascal

```
procedure KeyUp(var Key: Word; Shift: TShiftState); override;
```

C++

```
virtual __fastcall KeyUp(Word Key, TShiftState Shift);
```

Description

After calling the inherited method, KeyUp makes sure that selection using the Ctrl+Left and Ctrl+Right key combinations ends with an editable selection in the edit window. Derived classes can override this method to further supplement the processing that

takes place when the user releases a key.

See Also

CheckCursor (see page 1083)

KeyDown (see page 1087)

KeyPress (see page 1088)

SetSel (see page 1090)

29.1.2.3.14 TCustomMaskEdit.ReformatText Method

Applies the mask NewMask to the underlying text, and updates the internal representation of the mask.

Pascal

```
procedure ReformatText(const NewMask: string);
```

C++

```
__fastcall ReformatText(const UnicodeString NewMask);
```

Description

ReformatText does the bulk of the work of setting the EditMask (see page 1092) property at runtime. ReformatText should not be used in place of the EditMask (see page 1092) property, however, as it does not deal with the MaskState (see page 1096) property or the positioning of the cursor.

See Also

CheckCursor (see page 1083)

EditMask (see page 1092)

EditText (see page 1095)

MaskState (see page 1096)

Text (see page 1097)

29.1.2.3.15 TCustomMaskEdit.Reset Method

Returns the EditText (see page 1095) to its value at the time when the edit control last lost focus.

Pascal

```
procedure Reset; virtual;
```

C++

```
virtual __fastcall Reset();
```

Description

Reset cancels the most recent edits to the masked edit control. The EditText (see page 1095) returns to the value at the time the control last lost focus, not the time when it first regained focus. Thus, any changes to the EditText (see page 1095) that occurred programmatically while the edit control did not have focus are also discarded.

See Also

Clear (see page 1083)

EditText (see page 1095)

KeyPress (see page 1088)

29.1.2.3.16 TCustomMaskEdit.SetCursor Method

Moves the cursor to the first editable character at or before Pos, and selects that character.

Pascal

```
procedure SetCursor(Pos: Integer);
```

C++

```
__fastcall SetCursor(int Pos);
```

Description

SetCursor handles positioning of the cursor within the EditText (see page 1095) so that it is always on an editable character. It also ensures that at least one character is always selected. Because editing a masked edit control consists of replacing the blank characters in the mask, there must always be a selection to replace. The selection corresponds to one or more nonliteral characters in the mask, which are either blank or contain characters for the user to reenter.

See Also

CheckCursor (see page 1083)

GetPriorEditChar (see page 1086)

SetSel (see page 1090)

29.1.2.3.17 TCustomMaskEdit.SetSel Method

Selects character range.

Pascal

```
procedure SetSel(SelStart: Integer; SelStop: Integer);
```

C++

```
__fastcall SetSel(int SelStart, int SelStop);
```

Description

SetSel selects the indicated range of characters in the EditText (see page 1095), regardless of whether any of them are editable. If SelStart is the same as SelStop, SetSel positions the cursor at the SelStart character. When using SetSel, leave the cursor in an editable position if the intent is to create a selection for the user to replace.

SetSel can be used to select non-editable characters that can then be captured by using the SelText property or by copying to the Clipboard. Once the selection has been captured, return the cursor to an editable position by calling CheckCursor (see page 1083).

See Also

CheckCursor (see page 1083)

EditText (see page 1095)

GetSel (see page 1087)

SelText

SetCursor (see page 1090)

29.1.2.3.18 TCustomMaskEdit.Validate Method

Validates a specified string against the current mask.

Pascal

```
function Validate(const Value: string; var Pos: Integer): Boolean; virtual;
```

C++

```
virtual __fastcall Boolean Validate(const UnicodeString Value, int Pos);
```

Description

The ValidateEdit (see page 1091) method calls Validate to perform the actual validation of the current text against the edit mask. It returns true if the string specified by Value is consistent with the edit mask, false if it detects an invalid character. If ValidateEdit (see page 1091) returns false, it returns the position of the invalid character as the Pos parameter.

See Also

EditMask (see page 1092)

ValidateEdit (see page 1091)

29.1.2.3.19 TCustomMaskEdit.ValidateEdit Method

Validates the EditText (see page 1095) against the current mask.

Pascal

```
procedure ValidateEdit; virtual;
```

C++

```
virtual __fastcall ValidateEdit();
```

Description

ValidateEdit attempts to validate the EditText (see page 1095) after each keystroke, and when the user attempts to leave the masked edit field. If the EditText (see page 1095) does not match the specifications of the EditMask (see page 1092), ValidateEdit calls ValidateError (see page 1091) to raise an exception, and returns the cursor to the first position where the EditText (see page 1095) did not comply with the EditMask (see page 1092).

See Also

EditMask (see page 1092)

EditText (see page 1095)

KeyPress (see page 1088)

ValidateError (see page 1091)

29.1.2.3.20 TCustomMaskEdit.ValidateError Method

Beeps and raises an EDBEditError (see page 1080) exception.

Pascal

```
procedure ValidateError; virtual;
```

C++

```
virtual __fastcall ValidateError();
```

Description

ValidateEdit (see page 1091) calls ValidateError when the EditText (see page 1095) fails to comply with the EditMask (see page 1092).

See Also

EditMask (see page 1092)

EditText (see page 1095)

ValidateEdit (see page 1091)

29.1.2.4 TCustomMaskEdit Properties

The properties of the TCustomMaskEdit class are listed here.

Protected Properties

	Name	Description
	EditMask (see page 1092)	Specifies the mask that represents what text is valid for the masked edit control.
	MaskState (see page 1096)	Indicates state information about how the mask is being applied.
	MaxLength (see page 1096)	Specifies the maximum number of characters that can appear in the edit control.

Public Properties

	Name	Description
	EditText (see page 1095)	Specifies the text that appears in the window of the masked edit control.
R	IsMasked (see page 1096)	Indicates whether a mask exists for the edit control.
	Text (see page 1097)	Denotes the underlying text for the masked edit control before the mask has been applied.

29.1.2.4.1 TCustomMaskEdit.EditMask Property

Specifies the mask that represents what text is valid for the masked edit control.

Pascal

```
property EditMask: TEditMask;
```

C++

```
__property TEditMask EditMask;
```

Description

Use EditMask to restrict the characters a user can enter into the masked edit control to valid characters and formats. If the user attempts to enter an invalid character, the edit control does not accept the character. Validation is performed on a character-by-character basis by the ValidateEdit (see page 1091) method.

Setting EditMask to an empty string removes the mask.

C++ Examples:

```
/*
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
```

```
  using bit-wise AND operations and displayed in the third mask edit box.
  An additional test is made to assure that the values in the IP address
  and in the subnet mask are in the interval from 0 to 255 ,
  as this validation test cannot be accomplished
  using the edit mask property alone.
  */
  void __fastcall TForm1::FormCreate(TObject *Sender)
  {
    // initialize the maskedit boxes to contain the empty string
    MaskEdit1->Text = "";
    MaskEdit2->Text = "";
    MaskEdit3->Text = "";

    // use the Courier font in the maskedit boxes
    MaskEdit1->Font->Name = "Courier";
    MaskEdit2->Font->Name = "Courier";
    MaskEdit3->Font->Name = "Courier";

    // initialize the mask of each maskedit box
    // according to IP address coding conventions
    MaskEdit1->EditMask = "!099.099.099.099;1; ";
    MaskEdit2->EditMask = "!099.099.099.099;1; ";
    MaskEdit3->EditMask = "!099.099.099.099;1; ";

    MaskEdit2->OnExit = MaskEdit2Exit;
  }

  void __fastcall TForm1::MaskEdit2Exit(TObject *Sender)
  {
    // extract the net and host address from the IP
    String IP = MaskEdit1->Text;
    int net1 = IP.SubString(0, 3).TrimRight().ToInt();
    int net2 = IP.SubString(5, 3).TrimRight().ToInt();
    int host1 = IP.SubString(9, 3).TrimRight().ToInt();
    int host2 = IP.SubString(13, 3).TrimRight().ToInt();

    // a range test that you cannot validate through edit masks
    if (net1 < 0 || net1 > 255 ||
        net2 < 0 || net2 > 255 ||
        host1 < 0 || host1 > 255 ||
        host2 < 0 || host2 > 255)
      throw(Exception("Not a valid IP address."));

    // extract the net and host mask from the subnet mask
    String mask = MaskEdit2->Text;
    int netmask1 = mask.SubString(0, 3).TrimRight().ToInt();
    int netmask2 = mask.SubString(5, 3).TrimRight().ToInt();
    int hostmask1 = mask.SubString(9, 3).TrimRight().ToInt();
    int hostmask2 = mask.SubString(13, 3).TrimRight().ToInt();

    // a range test that you cannot validate through edit masks
    if (netmask1 < 0 || netmask1 > 255 ||
        netmask2 < 0 || netmask2 > 255 ||
        hostmask1 < 0 || hostmask1 > 255 ||
        hostmask2 < 0 || hostmask2 > 255)
      throw(Exception("Not a valid subnet mask."));

    // compute the subnet address using bit-wise AND
    int sub_net1 = net1 & netmask1;
    int sub_net2 = net2 & netmask2;
    int sub_host1 = host1 & hostmask1;
    int sub_host2 = host2 & hostmask2;

    // display the subnet address
    MaskEdit3->Text =
      String(sub_net1) + "." + String(sub_net2) + "." +
```

```
    String(sub_host1) + "." + String(sub_host2);
}
```

Delphi Examples:

```
{
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255,
as this validation test cannot be accomplished
using the edit mask property alone.
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  { initialize the maskedit boxes to contain the empty string }
  MaskEdit1.Text := '';
  MaskEdit2.Text := '';
  MaskEdit3.Text := '';

  { use the Courier font in the maskedit boxes }
  MaskEdit1.Font.Name := 'Courier';
  MaskEdit2.Font.Name := 'Courier';
  MaskEdit3.Font.Name := 'Courier';

  {
  initialize the mask of each maskedit box
  according to IP address coding conventions
  }
  MaskEdit1.EditMask := '!099.099.099.099;1; ';
  MaskEdit2.EditMask := '!099.099.099.099;1; ';
  MaskEdit3.EditMask := '!099.099.099.099;1; ';

  MaskEdit2.OnExit := MaskEdit2Exit;
end;

procedure TForm1.MaskEdit2Exit(Sender: TObject);
var
  net1, net2, host1, host2,
  netmask1, netmask2, hostmask1, hostmask2,
  sub_net1, sub_net2, sub_host1, sub_host2: Integer;

  IP, mask: String;

begin
  // extract the net and host address from the IP
  IP := MaskEdit1.Text;
  net1 := StrToInt(TrimRight(Copy(IP, 0, 3)));
  net2 := StrToInt(TrimRight(Copy(IP, 5, 3)));
  host1 := StrToInt(TrimRight(Copy(IP, 9, 3)));
  host2 := StrToInt(TrimRight(Copy(IP, 13, 3)));

  // a range test that you cannot validate through edit masks
  if ((net1 < 0) Or (net1 > 255) Or
      (net2 < 0) Or (net2 > 255) Or
      (host1 < 0) Or (host1 > 255) Or
      (host2 < 0) Or (host2 > 255)) then
    raise EArgumentException.Create('Not a valid IP address.');

  // extract the net and host mask from the subnet mask
```

```
    mask := MaskEdit2.Text;
    netmask1 := StrToInt(TrimRight(Copy(mask, 0, 3)));
    netmask2 := StrToInt(TrimRight(Copy(mask, 5, 3)));
    hostmask1 := StrToInt(TrimRight(Copy(mask, 9, 3)));
    hostmask2 := StrToInt(TrimRight(Copy(mask, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((netmask1 < 0) Or (netmask1 > 255) Or
        (netmask2 < 0) Or (netmask2 > 255) Or
        (hostmask1 < 0) Or (hostmask1 > 255) Or
        (hostmask2 < 0) Or (hostmask2 > 255)) then
      raise EArgumentException.Create('Not a valid subnet mask.');

    // compute the subnet address using bit-wise AND
    sub_net1 := net1 And netmask1;
    sub_net2 := net2 And netmask2;
    sub_host1 := host1 And hostmask1;
    sub_host2 := host2 And hostmask2;

    // display the subnet address
    MaskEdit3.Text :=
      IntToStr(sub_net1) + '.' + IntToStr(sub_net2) + '.' +
      IntToStr(sub_host1) + '.' + IntToStr(sub_host2);
  end;
```

See Also

EditText (see page 1095)

IsMasked (see page 1096)

MaskState (see page 1096)

Text (see page 1097)

ValidateEdit (see page 1091)

ValidateError (see page 1091)

29.1.2.4.2 TCustomMaskEdit.EditText Property

Specifies the text that appears in the window of the masked edit control.

Pascal

```
property EditText: string;
```

C++

```
__property UnicodeString EditText;
```

Description

EditText is the value of the text for the edit control after it is formatted using the EditMask (see page 1092). Before the text is fully entered into a masked edit control, the EditText includes a blank character for each unentered character. As characters are entered, the blank characters in the EditText are replaced by the values that the user types.

If literal mask characters are included in the text, and no character is substituted for blanks, the values of EditText and Text (see page 1097) are the same.

Note: EditText only applies at runtime.

See Also

EditMask (see page 1092)

Text (see page 1097)

29.1.2.4.3 TCustomMaskEdit.IsMasked Property

Indicates whether a mask exists for the edit control.

Pascal

```
property IsMasked: Boolean;
```

C++

```
__property Boolean IsMasked;
```

Description

Read IsMasked to determine whether the EditMask (see page 1092) property has been set. If IsMasked is true, the control has a mask, and input is validated at runtime. If IsMasked is false, the control does not restrict editing at runtime.

See Also

EditMask (see page 1092)

29.1.2.4.4 TCustomMaskEdit.MaskState Property

Indicates state information about how the mask is being applied.

Pascal

```
property MaskState: TMaskedState;
```

C++

```
__property TMaskedState MaskState;
```

Description

MaskState is used internally to determine appropriate actions based on how the mask is currently being applied. MaskState is a set drawn from among the following values:

Value	Meaning
msMasked	A mask has been assigned. MaskState includes msMasked when IsMasked (see page 1096) is true.
msReEnter	An invalid character has been rejected and the next edit should be a valid replacement character.
msDBSetText	The text comes from a database field and should not be validated.

MaskState is exposed as a protected property so that derived classes can maintain and use this state information in the implementation of properties and methods.

See Also

IsMasked (see page 1096)

ValidateEdit (see page 1091)

29.1.2.4.5 TCustomMaskEdit.MaxLength Property

Specifies the maximum number of characters that can appear in the edit control.

Pascal

```
property MaxLength: Integer;
```

C++

```
__property int MaxLength;
```

Description

MaxLength is the length of the EditText (see page 1095). Set MaxLength to limit the number of characters that can appear in the edit control when there is no EditMask (see page 1092). If there is an EditMask (see page 1092), MaxLength is implied by the mask itself, and cannot be changed. The EditText (see page 1095) contains blank characters for each character that has not been entered, padding the end or beginning of every optional section, so it remains constant in length.

See Also

EditMask (see page 1092)

EditText (see page 1095)

IsMasked (see page 1096)

Text (see page 1097)

29.1.2.4.6 TCustomMaskEdit.Text Property

Denotes the underlying text for the masked edit control before the mask has been applied.

Pascal

```
property Text: TMaskedText;
```

C++

```
__property TMaskedText Text;
```

Description

Use Text to determine the underlying value of the text before it has been formatted by the mask. Text may differ from the EditText (see page 1095) visible in the masked edit control if the mask specifies that literal characters should be removed, if the mask includes spaces, or if the mask includes characters that have not yet been filled in by the user. Text will not contain the blank characters, spaces will not be replaced by _, and literal characters in the mask will be removed if the mask indicates they should be.

When setting Text, the value of the text is formatted using the mask, and the resulting EditText (see page 1095) is used to update the window.

If there is no mask, Text is the text that appears in the edit control.

C++ Examples:

```
/*
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255 ,
as this validation test cannot be accomplished
using the edit mask property alone.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
```

```
  // initialize the maskedit boxes to contain the empty string
  MaskEdit1->Text = "";
  MaskEdit2->Text = "";
  MaskEdit3->Text = "";

  // use the Courier font in the maskedit boxes
  MaskEdit1->Font->Name = "Courier";
  MaskEdit2->Font->Name = "Courier";
  MaskEdit3->Font->Name = "Courier";

  // initialize the mask of each maskedit box
  // according to IP address coding conventions
  MaskEdit1->EditMask = "!099.099.099.099;1; ";
  MaskEdit2->EditMask = "!099.099.099.099;1; ";
  MaskEdit3->EditMask = "!099.099.099.099;1; ";

  MaskEdit2->OnExit = MaskEdit2Exit;
}

void __fastcall TForm1::MaskEdit2Exit(TObject *Sender)
{
  // extract the net and host address from the IP
  String IP = MaskEdit1->Text;
  int net1 = IP.SubString(0, 3).TrimRight().ToInt();
  int net2 = IP.SubString(5, 3).TrimRight().ToInt();
  int host1 = IP.SubString(9, 3).TrimRight().ToInt();
  int host2 = IP.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (net1 < 0 || net1 > 255 ||
      net2 < 0 || net2 > 255 ||
      host1 < 0 || host1 > 255 ||
      host2 < 0 || host2 > 255)
    throw(Exception("Not a valid IP address."));

  // extract the net and host mask from the subnet mask
  String mask = MaskEdit2->Text;
  int netmask1 = mask.SubString(0, 3).TrimRight().ToInt();
  int netmask2 = mask.SubString(5, 3).TrimRight().ToInt();
  int hostmask1 = mask.SubString(9, 3).TrimRight().ToInt();
  int hostmask2 = mask.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (netmask1 < 0 || netmask1 > 255 ||
      netmask2 < 0 || netmask2 > 255 ||
      hostmask1 < 0 || hostmask1 > 255 ||
      hostmask2 < 0 || hostmask2 > 255)
    throw(Exception("Not a valid subnet mask."));

  // compute the subnet address using bit-wise AND
  int sub_net1 = net1 & netmask1;
  int sub_net2 = net2 & netmask2;
  int sub_host1 = host1 & hostmask1;
  int sub_host2 = host2 & hostmask2;

  // display the subnet address
  MaskEdit3->Text =
    String(sub_net1) + "." + String(sub_net2) + "." +
    String(sub_host1) + "." + String(sub_host2);
}
```

Delphi Examples:

```
{
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
```

```
  to contain the empty string, to use the Courier font and to
  use a particular mask, corresponding to IP address coding convetions.
  The mask edit boxes display the IP address, the subnet mask
  and the subnet address correspondingly. When the user exits the
  subnet mask edit box, the subnet address is automatically calculated
  using bit-wise AND operations and displayed in the third mask edit box.
  An additional test is made to assure that the values in the IP address
  and in the subnet mask are in the interval from 0 to 255,
  as this validation test cannot be accomplished
  using the edit mask property alone.
  }
  procedure TForm1.FormCreate(Sender: TObject);
  begin
    { initialize the maskedit boxes to contain the empty string }
    MaskEdit1.Text := '';
    MaskEdit2.Text := '';
    MaskEdit3.Text := '';

    { use the Courier font in the maskedit boxes }
    MaskEdit1.Font.Name := 'Courier';
    MaskEdit2.Font.Name := 'Courier';
    MaskEdit3.Font.Name := 'Courier';

    {
    initialize the mask of each maskedit box
    according to IP address coding conventions
    }
    MaskEdit1.EditMask := '!099.099.099.099;1; ';
    MaskEdit2.EditMask := '!099.099.099.099;1; ';
    MaskEdit3.EditMask := '!099.099.099.099;1; ';

    MaskEdit2.OnExit := MaskEdit2Exit;
  end;

  procedure TForm1.MaskEdit2Exit(Sender: TObject);
  var
    net1, net2, host1, host2,
    netmask1, netmask2, hostmask1, hostmask2,
    sub_net1, sub_net2, sub_host1, sub_host2: Integer;

    IP, mask: String;

  begin
    // extract the net and host address from the IP
    IP := MaskEdit1.Text;
    net1 := StrToInt(TrimRight(Copy(IP, 0, 3)));
    net2 := StrToInt(TrimRight(Copy(IP, 5, 3)));
    host1 := StrToInt(TrimRight(Copy(IP, 9, 3)));
    host2 := StrToInt(TrimRight(Copy(IP, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((net1 < 0) Or (net1 > 255) Or
        (net2 < 0) Or (net2 > 255) Or
        (host1 < 0) Or (host1 > 255) Or
        (host2 < 0) Or (host2 > 255)) then
      raise EArgumentException.Create('Not a valid IP address.');

    // extract the net and host mask from the subnet mask
    mask := MaskEdit2.Text;
    netmask1 := StrToInt(TrimRight(Copy(mask, 0, 3)));
    netmask2 := StrToInt(TrimRight(Copy(mask, 5, 3)));
    hostmask1 := StrToInt(TrimRight(Copy(mask, 9, 3)));
    hostmask2 := StrToInt(TrimRight(Copy(mask, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((netmask1 < 0) Or (netmask1 > 255) Or
```

```
        (netmask2 < 0) Or (netmask2 > 255) Or
        (hostmask1 < 0) Or (hostmask1 > 255) Or
        (hostmask2 < 0) Or (hostmask2 > 255)) then
      raise EArgumentException.Create('Not a valid subnet mask.');

    // compute the subnet address using bit-wise AND
    sub_net1 := net1 And netmask1;
    sub_net2 := net2 And netmask2;
    sub_host1 := host1 And hostmask1;
    sub_host2 := host2 And hostmask2;

    // display the subnet address
    MaskEdit3.Text :=
      IntToStr(sub_net1) + '.' + IntToStr(sub_net2) + '.' +
      IntToStr(sub_host1) + '.' + IntToStr(sub_host2);
  end;
```

See Also

EditMask (see page 1092)

EditText (see page 1095)

GetTextLen (see page 1087)

IsMasked (see page 1096)

ReformatText (see page 1089)

29.1.3 TMaskEdit Class

TMaskEdit implements a generic masked edit control.

Class Hierarchy

Pascal

```
TMaskEdit = class(TCustomMaskEdit);
```

C++

```
class TMaskEdit : public TCustomMaskEdit;
```

File

Mask

Description

Use a TMaskEdit object to put a masked edit control on your form. Masked edit controls validate the text the user enters against a mask that encodes the valid forms the text can take. The mask can also format text that is displayed to the user.

TMaskEdit implements the generic behavior introduced in TCustomMaskEdit (see page 1080). TMaskEdit publishes many of the properties and methods inherited from TCustomMaskEdit (see page 1080), but does not introduce any new behavior.

See Also

TDBEdit

TEdit

TCustomMaskEdit (see page 1080)

Edit Controls

29.1.3.1 TMaskEdit Members

The following tables list the members exposed by TMaskEdit.

Protected Methods

	Name	Description
	CheckCursor (see page 1083)	Moves the cursor to the closest editable character and selects it if nothing is currently selected.
V	EditCanModify (see page 1084)	Permits the user to edit control contents.
	GetFirstEditChar (see page 1084)	Returns the index in the EditMask (see page 1092) of the first nonliteral character.
	GetLastEditChar (see page 1085)	Returns the index in the EditMask (see page 1092) of the last nonliteral character.
	GetMaxChars (see page 1085)	Returns the length of the EditText (see page 1095).
	GetNextEditChar (see page 1086)	Returns the first index of a nonliteral character equal to or greater than an offset.
	GetPriorEditChar (see page 1086)	Returns the first index of a nonliteral character equal to or less than an offset.
	GetSel (see page 1087)	Returns the endpoints of the current selection in the EditText (see page 1095).
V	KeyDown (see page 1087)	Supplements the inherited KeyDown to process navigation keys and the Delete key.
V	KeyPress (see page 1088)	Supplements the inherited method to process the Esc and Enter keys.
V	KeyUp (see page 1088)	Key up event dispatcher.
	ReformatText (see page 1089)	Applies the mask NewMask to the underlying text, and updates the internal representation of the mask.
V	Reset (see page 1089)	Returns the EditText (see page 1095) to its value at the time when the edit control last lost focus.
	SetCursor (see page 1090)	Moves the cursor to the first editable character at or before Pos, and selects that character.
	SetSel (see page 1090)	Selects character range.
V	Validate (see page 1091)	Validates a specified string against the current mask.
V	ValidateError (see page 1091)	Beeps and raises an EDBEditError (see page 1080) exception.

Protected Properties

	Name	Description
	MaskState (see page 1096)	Indicates state information about how the mask is being applied.

Public Methods

	Name	Description
V	Clear (see page 1083)	Sets the Text (see page 1097) property to an empty string.
V	Create (see page 1082)	Creates an instance of TCustomMaskEdit (see page 1080).
	GetTextLen (see page 1087)	Returns the length of the underlying text.
V	ValidateEdit (see page 1091)	Validates the EditText (see page 1095) against the current mask.

Public Properties

	Name	Description
	EditText (see page 1095)	Specifies the text that appears in the window of the masked edit control.
R	IsMasked (see page 1096)	Indicates whether a mask exists for the edit control.

Published Events

	Name	Description
	OnChange (see page 1142)	Occurs when the text for the edit control may have changed.
	OnClick (see page 1142)	Occurs when the user clicks the control.
	OnDblClick (see page 1144)	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.
	OnDragDrop (see page 1144)	Occurs when the user drops an object being dragged.

	OnDragOver (see page 1147)	Occurs when the user drags an object over a control.
	OnEndDock (see page 1150)	Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.
	OnEndDrag (see page 1151)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.
	OnEnter (see page 1152)	Occurs when a control receives the input focus.
	OnExit (see page 1154)	Occurs when the input focus shifts away from one control to another.
	OnKeyDown (see page 1158)	Occurs when a user presses any key while the control has focus.
	OnKeyPress (see page 1162)	Occurs when a key is pressed.
	OnKeyUp (see page 1163)	Occurs when the user releases a key that has been pressed.
	OnMouseActivate (see page 1165)	Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.
	OnMouseDown (see page 1166)	Occurs when the user presses a mouse button with the mouse pointer over a control.
	OnMouseEnter (see page 1168)	Occurs when the user moves the mouse into a control.
	OnMouseLeave (see page 1168)	Occurs when the user moves the mouse outside of a control.
	OnMouseMove (see page 1168)	Occurs when the user moves the mouse pointer while the mouse pointer is over a control.
	OnMouseUp (see page 1171)	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.
	OnStartDock (see page 1173)	Occurs when the user begins to drag a control with a DragKind of dkDock.
	OnStartDrag (see page 1173)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

Published Properties

	Name	Description
	Align (see page 1104)	Determines how the control aligns within its container (parent control).
	Alignment (see page 1107)	Determines how the text is aligned within the text edit control.
	Anchors (see page 1108)	Specifies how the control is anchored to its parent.
	AutoSelect (see page 1108)	Determines whether all the text in the edit control is automatically selected when the control gets focus.
	AutoSize (see page 1109)	Determines whether the height of the edit control automatically resizes to accommodate the text.
	BevelEdges (see page 1110)	Specifies which edges of the control are beveled.
	BevelInner (see page 1110)	Specifies the cut of the inner bevel.
	BevelKind (see page 1111)	Specifies the control's bevel style.
	BevelOuter (see page 1111)	Specifies the cut of the outer bevel.
	BevelWidth (see page 1112)	Specifies the width of the inner and outer bevels.
	BiDiMode (see page 1112)	Specifies the bi-directional mode for the control.
	BorderStyle (see page 1112)	Determines whether the edit control has a single line border around the client area.
	CharCase (see page 1113)	Determines the case of the text within the edit control.
	Color (see page 1113)	Specifies the background color of the control.
	Constraints (see page 1114)	Specifies the size constraints for the control.
	Ctl3D (see page 1115)	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
	DoubleBuffered (see page 1115)	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
	DragCursor (see page 1117)	Indicates the image used to represent the mouse pointer when the control is being dragged.
	DragKind (see page 1117)	Specifies whether the control is being dragged normally or for docking.
	DragMode (see page 1118)	Determines how the control initiates drag-and-drop or drag-and-dock operations.
	EditMask (see page 1119)	Specifies the mask that represents what text is valid for the masked edit control.
	Enabled (see page 1122)	Controls whether the control responds to mouse, keyboard, and timer events.
	Font (see page 1123)	Controls the attributes of text written on or in the control.
	ImeMode (see page 1124)	Determines the behavior of the input method editor (IME).
	ImeName (see page 1124)	Specifies the input method editor (IME) to use for converting keyboard input to Asian language characters.
	MaxLength (see page 1125)	Specifies the maximum number of characters that can appear in the edit control.

	ParentBiDiMode (see page 1125)	Specifies whether the control uses its parent's BiDiMode.
	ParentColor (see page 1126)	Determines where a control looks for its color information.
	ParentCtl3D (see page 1126)	Determines where a component looks to determine if it should appear three dimensional.
	ParentDoubleBuffered (see page 1127)	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
	ParentFont (see page 1127)	Determines where a control looks for its font information.
	ParentShowHint (see page 1127)	Determines where a control looks to find out if its Help Hint should be shown.
	PasswordChar (see page 1128)	Indicates the character, if any, to display in place of the actual characters typed in the control.
	PopupMenu (see page 1128)	Identifies the pop-up menu associated with the control.
	ReadOnly (see page 1133)	Determines whether the user can change the text of the edit control.
	ShowHint (see page 1134)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	TabOrder (see page 1135)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 1136)	Determines if the user can tab to a control.
	Text (see page 1136)	Denotes the underlying text for the masked edit control before the mask has been applied.
	TextHint (see page 1139)	A hint or message to be displayed when the Text property is empty.
	Visible (see page 1139)	Determines whether the component appears onscreen.

29.1.3.2 TMaskEdit Properties

The properties of the TMaskEdit class are listed here.

Published Properties

	Name	Description
	Align (see page 1104)	Determines how the control aligns within its container (parent control).
	Alignment (see page 1107)	Determines how the text is aligned within the text edit control.
	Anchors (see page 1108)	Specifies how the control is anchored to its parent.
	AutoSelect (see page 1108)	Determines whether all the text in the edit control is automatically selected when the control gets focus.
	AutoSize (see page 1109)	Determines whether the height of the edit control automatically resizes to accommodate the text.
	BevelEdges (see page 1110)	Specifies which edges of the control are beveled.
	BevelInner (see page 1110)	Specifies the cut of the inner bevel.
	BevelKind (see page 1111)	Specifies the control's bevel style.
	BevelOuter (see page 1111)	Specifies the cut of the outer bevel.
	BevelWidth (see page 1112)	Specifies the width of the inner and outer bevels.
	BiDiMode (see page 1112)	Specifies the bi-directional mode for the control.
	BorderStyle (see page 1112)	Determines whether the edit control has a single line border around the client area.
	CharCase (see page 1113)	Determines the case of the text within the edit control.
	Color (see page 1113)	Specifies the background color of the control.
	Constraints (see page 1114)	Specifies the size constraints for the control.
	Ctl3D (see page 1115)	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
	DoubleBuffered (see page 1115)	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
	DragCursor (see page 1117)	Indicates the image used to represent the mouse pointer when the control is being dragged.
	DragKind (see page 1117)	Specifies whether the control is being dragged normally or for docking.
	DragMode (see page 1118)	Determines how the control initiates drag-and-drop or drag-and-dock operations.
	EditMask (see page 1119)	Specifies the mask that represents what text is valid for the masked edit control.
	Enabled (see page 1122)	Controls whether the control responds to mouse, keyboard, and timer events.
	Font (see page 1123)	Controls the attributes of text written on or in the control.
	ImeMode (see page 1124)	Determines the behavior of the input method editor (IME).

	ImeName (see page 1124)	Specifies the input method editor (IME) to use for converting keyboard input to Asian language characters.
	MaxLength (see page 1125)	Specifies the maximum number of characters that can appear in the edit control.
	ParentBiDiMode (see page 1125)	Specifies whether the control uses its parent's BiDiMode.
	ParentColor (see page 1126)	Determines where a control looks for its color information.
	ParentCtl3D (see page 1126)	Determines where a component looks to determine if it should appear three dimensional.
	ParentDoubleBuffered (see page 1127)	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
	ParentFont (see page 1127)	Determines where a control looks for its font information.
	ParentShowHint (see page 1127)	Determines where a control looks to find out if its Help Hint should be shown.
	PasswordChar (see page 1128)	Indicates the character, if any, to display in place of the actual characters typed in the control.
	PopupMenu (see page 1128)	Identifies the pop-up menu associated with the control.
	ReadOnly (see page 1133)	Determines whether the user can change the text of the edit control.
	ShowHint (see page 1134)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	TabOrder (see page 1135)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 1136)	Determines if the user can tab to a control.
	Text (see page 1136)	Denotes the underlying text for the masked edit control before the mask has been applied.
	TextHint (see page 1139)	A hint or message to be displayed when the Text property is empty.
	Visible (see page 1139)	Determines whether the component appears onscreen.

29.1.3.2.1 TMaskEdit.Align Property

Determines how the control aligns within its container (parent control).

Pascal

```
property Align;
```

C++

```
__property Align;
```

Description

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes. When the parent is resized, an aligned control also resizes so that it continues to span the top, bottom, left, or right edge of the parent.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel.

Tip: If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

Note: To cause a control to maintain a specified relationship with an edge of its parent, but not necessarily lie along one edge of the parent, use the Anchors property instead.

C++ Examples:

```
/*
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
*/

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  TSplitter* ps = new TSplitter(Form1); // These objects will all be cleaned up by the owner
(Form1).
  TFileListBox* pflb = new TFileListBox(Form1); // The parent (Form1) will clean up these
objects.
  TDirectoryListBox* pdlb = new TDirectoryListBox(Form1);

  // Line up the directory list box on the left of the form.
  pdlb->Parent = Form1;
  pdlb->Align = alLeft;
  pdlb->Width = Form1->ClientWidth/3;

  // Now use the splitter to divide the directory pane from the file pane.
  ps->Parent = Form1;
  ps->Left = pdlb->Left + pdlb->Width + 1; // Move to right of directory list.
  ps->Align = pdlb->Align; // Give same alignment as directory list.

  // Each pane must be at least one quarter of the form's width.
  ps->MinSize = Form1->ClientWidth/4;

  // Finally, create the last pane - a file list box.
  pflb->Parent = Form1;
  pflb->Align = alClient;
  pdlb->FileList = pflb; // Link the file list box to the directory list box.
}
/*
Dynamically create a TProgressBar control and align it to
the bottom of the form.
*/

#include <ComCtrls.hpp>
#include <Controls.hpp>

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TProgressBar *ProgressBar = new TProgressBar(this);  // The owner will clean this up.
  ProgressBar->Parent = this;
  ProgressBar->Align = alBottom;
  TButton *Button1 = new TButton(this);  // The owner will clean this up.
  Button1->Parent = this;
}
/*
This example dynamically creates a Page Control, then a
series of Tab Sheets on the Page Control.
*/
#include <Comctrls.hpp>
TPageControl* ppc;
const int MAXTABS = 3;
TTabSheet* pts[MAXTABS];
const char * ppcTabTitles[] = { "ShortString", "Orders", "Items", "Parts" };
int iTabTitles = sizeof(ppcTabTitles)/sizeof(ppcTabTitles[0]);
const TColor colorPalette[12] = {
  clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia,
  clTeal, clNavy, clMaroon, clLime, clOlive, clPurple};
```

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    ppc = new TPageControl(this); // The owner (this) will clean this up.
    ppc->Parent = this;
    ppc->Align = alClient;
    for (int i=0;i<iTabTitles;i++)
  {
        pts[i] = new TTabSheet(this); // The owner (this) will clean this up.
        pts[i]->PageControl = ppc;
        pts[i]->Name = AnsiString("pts") + ppcTabTitles[i];
        pts[i]->Caption = ppcTabTitles[i];
        pts[i]->Brush->Color = colorPalette[i];
  }
}
```

Delphi Examples:

```
{
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  Split: TSplitter;
  Files: TFileListBox;
  Dirs: TDirectoryListBox;
begin
  { first add a directory list box to the form }
  Dirs := TDirectoryListBox.Create(Form1);
  { line it up on the left of the form }
  Dirs.Parent := Form1;
  Dirs.Align := alLeft;
  Dirs.Width := Form1.ClientWidth div 3;
{ now add the splitter to divide the directory pane from the file pane }
  Split := TSplitter.Create(Form1);
  Split.Parent := Form1;
  { make sure the splitter is to the right of the directory list box! }
  Split.Left := Dirs.Left + Dirs.Width + 1;
  Split.Align := Dirs.Align; { give it the same alignment as the directory }
  { each pane must be at least one quarter of the form's width }
  Split.MinSize := Form1.ClientWidth div 4;
  { Finally, create the last pane - a file list box }
  Files := TFileListBox.Create(Form1);
  Files.Parent := Form1;
  Files.Align := alClient;
  Dirs.FileList := Files; { link the file list box to the directory list box }
end;
{
Dynamically create a TProgressBar control and align it to
the bottom of the form.
}

var
  I: Integer;

procedure TForm1.Button1Click(Sender: TObject);
var
  ProgressBar: TProgressBar;
begin
  ProgressBar := TProgressBar.Create(Self);
```

```
    with ProgressBar do
    begin
      Parent := Self;
      Align := alBottom;
    end;
  end;
  {
  This example dynamically creates a Page Control, then a
  series of Tab Sheets on the Page Control.
  }
  var
    PageControl1: TPageControl;
    pts: array[0..3] of TTabSheet;
  const
    TabTitles: array[0..3] of ShortString = ('Customer', 'Orders', 'Items', 'Parts' );

  procedure TForm1.FormCreate(Sender: TObject);
  const
    colorpalette: Array[0..11] of TColor = (
      clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia, clTeal,
      clNavy, clMaroon, clLime, clOlive, clPurple);
  var
    i: Integer;
  begin
    PageControl1 := TPageControl.Create(Self);
    PageControl1.Parent := Self;
    PageControl1.Align := alClient;
    for i := Low(TabTitles) to High(TabTitles) do
    begin
      pts[i]:= TTabSheet.Create(PageControl1);
      with pts[i] do
      begin
        PageControl := PageControl1;
        Name := 'ts' + TabTitles[i];
        Caption := TabTitles[i];
        Brush.Color := colorPalette[i];
     end;
    end;
  end;

  procedure TForm1.FormDestroy(Sender: TObject);
  var
    i: Integer;
  begin
    for i := Low(TabTitles) to High(TabTitles) do
    begin
      pts[i].Free;
    end;
    PageControl1.Free;
  end;
```

See Also

Parent

Anchors

29.1.3.2.2 TMaskEdit.Alignment Property

Determines how the text is aligned within the text edit control.

Pascal

```
property Alignment;
```

C++

```
__property Alignment;
```

Description

Use Alignment to change the way the text is formatted in the text edit control. Alignment can take one of the following values:

Value	Meaning
taLeftJustify	Align text to the left side of the control
taCenter	Center text horizontally in the control.
taRightJustify	Align text to the right side of the control.

See Also

SetAlignment

Text

Displaying and Editing Fields in an Edit Box

29.1.3.2.3 TMaskEdit.Anchors Property

Specifies how the control is anchored to its parent.

Pascal

```
property Anchors;
```

C++

```
__property Anchors;
```

Description

Use Anchors to ensure that a control maintains its current position relative to an edge of its parent, even if the parent is resized. When its parent is resized, the control holds its position relative to the edges to which it is anchored.

If a control is anchored to opposite edges of its parent, the control stretches when its parent is resized. For example, if a control has its Anchors property set to [akLeft, akRight], the control stretches when the width of its parent changes.

Anchors is enforced only when the parent is resized. Thus, for example, if a control is anchored to opposite edges of a form at design time and the form is created in a maximized state, the control is not stretched because the form is not resized after the control is created.

Note: If a control should maintain contact with three edges of its parent (hugging one side of the parent and stretching the length of that side), use the Align property instead. Unlike Anchors, Align allows controls to adjust to changes in the size of other aligned sibling controls as well as changes to the parent's size.

See Also

Parent

Align

29.1.3.2.4 TMaskEdit.AutoSelect Property

Determines whether all the text in the edit control is automatically selected when the control gets focus.

Pascal

```
property AutoSelect;
```

C++

```
__property AutoSelect;
```

Description

Set AutoSelect to select all the text when the edit control gets focus. AutoSelect only applies to single-line edit controls.

Use AutoSelect when the user is more likely to replace the text in the edit control than to append to it.

See Also

SelectAll

SelLength

SelStart

SelText

SetSelTextBuf

29.1.3.2.5 TMaskEdit.AutoSize Property

Determines whether the height of the edit control automatically resizes to accommodate the text.

Pascal

```
property AutoSize;
```

C++

```
__property AutoSize;
```

Description

Use AutoSize to make the edit control adjust its size automatically so the client area accommodates the height of the text. When AutoSize is false, the edit control has a fixed height. When AutoSize is true, the size of the control is readjusted whenever a change occurs that could affect the height of the control, such as a change to the font or border style.

Note: AutoSize only adjusts the height of the edit control. Even with AutoSize set to true, text in the edit control may appear truncated if it extends beyond the width of the control.

C++ Examples:

```
/*
This example uses an edit box and a button on a form. When
the user clicks the button, the font in the edit box
enlarges, and the edit box enlarges also to accommodate the
larger font size.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Edit1->AutoSize = true;
  Edit1->Font->Size = 20;
  Edit1->Text = "The edit box is taller now";
}
```

Delphi Examples:

```
{
{
This example uses an edit box and a button on a form. When
the user clicks the button, the font in the edit box
enlarges, and the edit box enlarges also to accommodate the
```

```
  larger font size.
  }
  procedure TForm1.Button1Click(Sender: TObject);
  begin
    Edit1.AutoSize := True;
    Edit1.Font.Size := 20;
    Edit1.Text := 'The edit box is taller now';
  end;
```

See Also

BorderStyle

ClientHeight

ControlStyle

Height

29.1.3.2.6 TMaskEdit.BevelEdges Property

Specifies which edges of the control are beveled.

Pascal

```
property BevelEdges;
```

C++

```
__property BevelEdges;
```

Description

Use BevelEdges to get or set which edges of the control are beveled. The BevelInner, BevelOuter, and BevelKind properties determine the appearance of the specified edges.

See Also

BevelInner

BevelOuter

BevelKind

29.1.3.2.7 TMaskEdit.BevelInner Property

Specifies the cut of the inner bevel.

Pascal

```
property BevelInner;
```

C++

```
__property BevelInner;
```

Description

Use BevelInner to specify whether the inner bevel has a raised, lowered, or flat look.

The inner bevel appears immediately inside the outer bevel. If there is no outer bevel (BevelOuter is bvNone), the inner bevel appears immediately inside the border.

See Also

BevelEdges

BevelOuter

BevelKind

BevelWidth

BorderWidth

29.1.3.2.8 TMaskEdit.BevelKind Property

Specifies the control's bevel style.

Pascal

```
property BevelKind;
```

C++

```
__property BevelKind;
```

Description

Use BevelKind to modify the appearance of a bevel. BevelKind influences how sharply the bevel stands out.

BevelKind, in combination with BevelWidth and the cut of the bevel specified by BevelInner or BevelOuter, can create a variety of effects. Experiment with various combinations to get the look you want.

See Also

BevelEdges

BevelOuter

BevelInner

BevelWidth

29.1.3.2.9 TMaskEdit.BevelOuter Property

Specifies the cut of the outer bevel.

Pascal

```
property BevelOuter;
```

C++

```
__property BevelOuter;
```

Description

Use BevelOuter to specify whether the outer bevel has a raised, lowered, or flat look.

The outer bevel appears immediately inside the border and outside the inner bevel.

See Also

BevelEdges

BevelInner

BevelKind

BevelWidth

29.1.3.2.10 TMaskEdit.BevelWidth Property

Specifies the width of the inner and outer bevels.

Pascal

```
property BevelWidth;
```

C++

```
__property BevelWidth;
```

Description

Use BevelWidth to specify the width, in pixels, of the inner and outer bevels.

See Also

BevelEdges

BevelInner

BevelKind

BevelOuter

29.1.3.2.11 TMaskEdit.BiDiMode Property

Specifies the bi-directional mode for the control.

Pascal

```
property BiDiMode;
```

C++

```
__property BiDiMode;
```

Description

Use BiDiMode to enable the control to adjust its appearance and behavior automatically when the application runs in a locale that reads from right to left instead of left to right. The bi-directional mode controls the reading order for the text, the placement of the vertical scroll bar, and whether the alignment is changed.

Alignment does not change for controls that are known to contain number, date, time, or currency values. For example, with data aware controls, the alignment does not change for the following field types: ftSmallint, ftInteger, ftWord, ftFloat, ftCurrency, ftBCD, ftDate, ftTime, ftDateTime, ftAutoInc.

See Also

UseRightToLeftAlignment

UseRightToLeftReading

UseRightToLeftScrollBar

Including Bi-directional Functionality in Applications

29.1.3.2.12 TMaskEdit.BorderStyle Property

Determines whether the edit control has a single line border around the client area.

Pascal

```
property BorderStyle;
```

C++

```
__property BorderStyle;
```

Description

Use BorderStyle to affect the sharpness with which the client area of the edit control stands out. BorderStyle can have a value of either bsSingle or bsNone. If BorderStyle is bsSingle, the edit control has a single-line border around the client area. If BorderStyle is bsNone, there will be no border.

29.1.3.2.13 TMaskEdit.CharCase Property

Determines the case of the text within the edit control.

Pascal

```
property CharCase;
```

C++

```
__property CharCase;
```

Description

Use CharCase to force the contents of the edit control to assume a particular case.

When CharCase is set to ecLowerCase or ecUpperCase, the case of characters is converted as the user types them into the edit control. Changing the CharCase property to ecLowerCase or ecUpperCase changes the actual contents of the text, not just the appearance. Any case information is lost and can't be recaptured by changing CharCase to ecNormal.

See Also

Text

29.1.3.2.14 TMaskEdit.Color Property

Specifies the background color of the control.

Pascal

```
property Color;
```

C++

```
__property Color;
```

Description

Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to false.

C++ Examples:

```
/*
The following code changes the color of a shape control
using the Color dialog box. The example displays the Color
dialog box when the Button1 button is clicked, allowing the
user to select a color with the dialog box. When a color
value is selected with the dialog box, it is assigned to
the shape control.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
```

```
{
  if (ColorDialog1->Execute())
    Shape1->Brush->Color = ColorDialog1->Color;
}
```

Delphi Examples:

```
{
The following code changes the color of a shape control
using the Color dialog box. The example displays the Color
dialog box when the Button1 button is clicked, allowing the
user to select a color with the dialog box. When a color
value is selected with the dialog box, it is assigned to
the shape control.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  if ColorDialog1.Execute then
    Shape1.Brush.Color := ColorDialog1.Color;
end;
```

See Also

ParentColor

29.1.3.2.15 TMaskEdit.Constraints Property

Specifies the size constraints for the control.

Pascal

```
property Constraints;
```

C++

```
__property Constraints;
```

Description

Use Constraints to specify the minimum and maximum width and height of the control. When Constraints contains maximum or minimum values, the control cannot be resized to violate those constraints.

Note: Constraints apply to the height of the form. However, the height of the form depends on how large Windows makes the title bar. For example, Windows XP uses much larger title bars than other windowing systems. To work around this, note the ClientHeight when you design your form and set the constraints in the FormCreate event. In the following example, ClientHeight is represented by x: `Constraints.MinHeight := x + Height - ClientHeight`.

Warning: Do not set up constraints that conflict with the value of the Align or Anchors property. When these properties conflict, the response of the control to resize attempts is not well-defined.

See Also

AutoSize

Align

Anchors

OnConstrainedResize

OnCanResize

29.1.3.2.16 TMaskEdit.Ctl3D Property

Determines whether a control has a three-dimensional (3-D) or two-dimensional look.

Pascal

```
property Ctl3D;
```

C++

```
__property Ctl3D;
```

Description

Ctl3D is provided for backward compatibility. It is not used by 32-bit versions of Windows or NT4.0 and later, with the exception of Windows XP Home, where it is used on occasion.

On earlier platforms, Ctl3D controlled whether the control had a flat or beveled appearance.

See Also

ParentCtl3D

BevelKind

BevelInner

BevelOuter

BorderWidth

29.1.3.2.17 TMaskEdit.DoubleBuffered Property

Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.

Pascal

```
property DoubleBuffered;
```

C++

```
__property DoubleBuffered;
```

Description

When DoubleBuffered is false, the windowed control paints itself directly to the window. When DoubleBuffered is true, the windowed control paints itself to an in-memory bitmap that is then used to paint the window. Double buffering reduces the amount of flicker when the control repaints, but is more memory intensive.

When a windowed control is a dock site and has an associated dock manager, it must be double-buffered.

Note: Some controls, such as TRichEdit, can't paint themselves into a bitmap. For such controls, DoubleBuffered must be set to false.

C++ Examples:

```
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/
```

```
// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
dropped label.
}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Not');
  ListBox1.Items.Add('In');
  ListBox1.Items.Add('Alphabetical');
  ListBox1.Items.Add('Order');
end;

// This OnDragOver event handler permits the list box to
// accept a dropped label:

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

// This OnDragDrop event handler implements the drop behavior.

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
```

```
      with Sender as TListBox do
      begin
        Font := (Source as TLabel).Font;
        Color := (Source as TLabel).Color;
        DoubleBuffered := true;
        Color := clWindow;
      end;
    end;
  end;
```

See Also

DockSite

DockManager

PaintWindow

29.1.3.2.18 TMaskEdit.DragCursor Property

Indicates the image used to represent the mouse pointer when the control is being dragged.

Pascal

```
property DragCursor;
```

C++

```
__property DragCursor;
```

Description

Use the DragCursor property to change the cursor image presented when the control is being dragged.

Note: To make a custom cursor available for the DragCursor property, see the Cursor property.

See Also

Cursor

DragKind

DragMode

29.1.3.2.19 TMaskEdit.DragKind Property

Specifies whether the control is being dragged normally or for docking.

Pascal

```
property DragKind;
```

C++

```
__property DragKind;
```

Description

Use DragKind to get or set whether the control participates in drag-and-drop operations, or drag-and-dock operations.

See Also

DragCursor

DragMode

29.1.3.2.20 TMaskEdit.DragMode Property

Determines how the control initiates drag-and-drop or drag-and-dock operations.

Pascal

```
property DragMode;
```

C++

```
__property DragMode;
```

Description

Use DragMode to control when the user can drag the control. Disable the drag-and-drop or drag-and-dock capability at runtime by setting the DragMode property value to dmManual. Enable automatic dragging by setting DragMode to dmAutomatic.

C++ Examples:

```
/*
The following code uses GetNodeAt to add a dragged node as a
child of the node under the mouse when it is dropped.  This
example requires a populated TreeView.  Also, the TTreeView
DragMode property set to dmAutomatic, and the TTreeView
OnDragOver event handler must be implemented to accept the
drop.
*/
void __fastcall TForm1::TreeView1DragDrop(TObject *Sender, TObject *Source,
      int X, int Y)
{
  if (Source != dynamic_cast<TObject *>(TreeView1) || TreeView1->Selected == NULL)
    return;
  THitTests HT = TreeView1->GetHitTestInfoAt(X, Y);
  TNodeAttachMode AttachMode;
  TTreeNode *Item = TreeView1->GetNodeAt(X, Y);
  if (HT.Contains(htOnItem) || HT.Contains(htOnIcon))
    AttachMode = naAddChild;
  else if (HT.Contains(htNowhere))
    AttachMode = naAdd;
  else if (HT.Contains(htOnIndent))
    AttachMode = naInsert;
  else
    return;
  TreeView1->Selected->MoveTo(Item, AttachMode);
}

void __fastcall TForm1::TreeView1DragOver(TObject *Sender, TObject *Source,
      int X, int Y, TDragState State, bool &Accept)
{
  Accept = (Source == dynamic_cast<TObject *>(TreeView1));
}
```

Delphi Examples:

```
{
The following code uses GetNodeAt to add a dragged node as a
child of the node under the mouse when it is dropped.  This
example requires a populated TreeView.  Also, the TTreeView
DragMode property must be set to dmAutomatic, and the TTreeView
OnDragOver event handler must be implemented to accept the
drop.
}
procedure TForm1.TreeView1DragDrop(Sender, Source: TObject; X, Y: Integer);
var
  AnItem: TTreeNode;
  AttachMode: TNodeAttachMode;
```

```
    HT: THitTests;
  begin
    if TreeView1.Selected = nil then Exit;
    HT := TreeView1.GetHitTestInfoAt(X, Y);
    AnItem := TreeView1.GetNodeAt(X, Y);
    if (HT - [htOnItem, htOnIcon, htNowhere, htOnIndent] <> HT) then
    begin
      if (htOnItem in HT) or (htOnIcon in HT) then
        AttachMode := naAddChild
      else if htNowhere in HT then AttachMode := naAdd
      else if htOnIndent in HT then AttachMode := naInsert;
      TreeView1.Selected.MoveTo(AnItem, AttachMode);
    end;
  end;

  procedure TForm1.TreeView1DragOver(Sender, Source: TObject; X, Y: Integer;
    State: TDragState; var Accept: Boolean);
  begin
    Accept := Source is TTreeView;
  end;
```

See Also

BeginDrag

DragCursor

EndDrag

OnDragDrop

OnDragOver

OnEndDrag

OnStartDrag

DragKind

Arranging the Order of Persistent Columns

29.1.3.2.21 TMaskEdit.EditMask Property

Specifies the mask that represents what text is valid for the masked edit control.

Pascal

```
property EditMask: TEditMask;
```

C++

```
__property TEditMask EditMask;
```

Description

Use EditMask to restrict the characters a user can enter into the masked edit control to valid characters and formats. If the user attempts to enter an invalid character, the edit control does not accept the character. Validation is performed on a character-by-character basis by the ValidateEdit method.

Setting EditMask to an empty string removes the mask.

C++ Examples:

```
/*
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
```

```
  to contain the empty string, to use the Courier font and to
  use a particular mask, corresponding to IP address coding convetions.
  The mask edit boxes display the IP address, the subnet mask
  and the subnet address correspondingly. When the user exits the
  subnet mask edit box, the subnet address is automatically calculated
  using bit-wise AND operations and displayed in the third mask edit box.
  An additional test is made to assure that the values in the IP address
  and in the subnet mask are in the interval from 0 to 255 ,
  as this validation test cannot be accomplished
  using the edit mask property alone.
  */
  void __fastcall TForm1::FormCreate(TObject *Sender)
  {
    // initialize the maskedit boxes to contain the empty string
    MaskEdit1->Text = "";
    MaskEdit2->Text = "";
    MaskEdit3->Text = "";

    // use the Courier font in the maskedit boxes
    MaskEdit1->Font->Name = "Courier";
    MaskEdit2->Font->Name = "Courier";
    MaskEdit3->Font->Name = "Courier";

    // initialize the mask of each maskedit box
    // according to IP address coding conventions
    MaskEdit1->EditMask = "!099.099.099.099;1; ";
    MaskEdit2->EditMask = "!099.099.099.099;1; ";
    MaskEdit3->EditMask = "!099.099.099.099;1; ";

    MaskEdit2->OnExit = MaskEdit2Exit;
  }

  void __fastcall TForm1::MaskEdit2Exit(TObject *Sender)
  {
    // extract the net and host address from the IP
    String IP = MaskEdit1->Text;
    int net1 = IP.SubString(0, 3).TrimRight().ToInt();
    int net2 = IP.SubString(5, 3).TrimRight().ToInt();
    int host1 = IP.SubString(9, 3).TrimRight().ToInt();
    int host2 = IP.SubString(13, 3).TrimRight().ToInt();

    // a range test that you cannot validate through edit masks
    if (net1 < 0 || net1 > 255 ||
        net2 < 0 || net2 > 255 ||
        host1 < 0 || host1 > 255 ||
        host2 < 0 || host2 > 255)
      throw(Exception("Not a valid IP address."));

    // extract the net and host mask from the subnet mask
    String mask = MaskEdit2->Text;
    int netmask1 = mask.SubString(0, 3).TrimRight().ToInt();
    int netmask2 = mask.SubString(5, 3).TrimRight().ToInt();
    int hostmask1 = mask.SubString(9, 3).TrimRight().ToInt();
    int hostmask2 = mask.SubString(13, 3).TrimRight().ToInt();

    // a range test that you cannot validate through edit masks
    if (netmask1 < 0 || netmask1 > 255 ||
        netmask2 < 0 || netmask2 > 255 ||
        hostmask1 < 0 || hostmask1 > 255 ||
        hostmask2 < 0 || hostmask2 > 255)
      throw(Exception("Not a valid subnet mask."));

    // compute the subnet address using bit-wise AND
    int sub_net1 = net1 & netmask1;
    int sub_net2 = net2 & netmask2;
    int sub_host1 = host1 & hostmask1;
```

```
  int sub_host2 = host2 & hostmask2;

  // display the subnet address
  MaskEdit3->Text =
    String(sub_net1) + "." + String(sub_net2) + "." +
    String(sub_host1) + "." + String(sub_host2);
}
```

Delphi Examples:

```
{
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255,
as this validation test cannot be accomplished
using the edit mask property alone.
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  { initialize the maskedit boxes to contain the empty string }
  MaskEdit1.Text := '';
  MaskEdit2.Text := '';
  MaskEdit3.Text := '';

  { use the Courier font in the maskedit boxes }
  MaskEdit1.Font.Name := 'Courier';
  MaskEdit2.Font.Name := 'Courier';
  MaskEdit3.Font.Name := 'Courier';

  {
  initialize the mask of each maskedit box
  according to IP address coding conventions
  }
  MaskEdit1.EditMask := '!099.099.099.099;1; ';
  MaskEdit2.EditMask := '!099.099.099.099;1; ';
  MaskEdit3.EditMask := '!099.099.099.099;1; ';

  MaskEdit2.OnExit := MaskEdit2Exit;
end;

procedure TForm1.MaskEdit2Exit(Sender: TObject);
var
  net1, net2, host1, host2,
  netmask1, netmask2, hostmask1, hostmask2,
  sub_net1, sub_net2, sub_host1, sub_host2: Integer;

  IP, mask: String;

begin
  // extract the net and host address from the IP
  IP := MaskEdit1.Text;
  net1 := StrToInt(TrimRight(Copy(IP, 0, 3)));
  net2 := StrToInt(TrimRight(Copy(IP, 5, 3)));
  host1 := StrToInt(TrimRight(Copy(IP, 9, 3)));
  host2 := StrToInt(TrimRight(Copy(IP, 13, 3)));

  // a range test that you cannot validate through edit masks
  if ((net1 < 0) Or (net1 > 255) Or
      (net2 < 0) Or (net2 > 255) Or
```

```
        (host1 < 0) Or (host1 > 255) Or
        (host2 < 0) Or (host2 > 255)) then
      raise EArgumentException.Create('Not a valid IP address.');

    // extract the net and host mask from the subnet mask
    mask := MaskEdit2.Text;
    netmask1 := StrToInt(TrimRight(Copy(mask, 0, 3)));
    netmask2 := StrToInt(TrimRight(Copy(mask, 5, 3)));
    hostmask1 := StrToInt(TrimRight(Copy(mask, 9, 3)));
    hostmask2 := StrToInt(TrimRight(Copy(mask, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((netmask1 < 0) Or (netmask1 > 255) Or
        (netmask2 < 0) Or (netmask2 > 255) Or
        (hostmask1 < 0) Or (hostmask1 > 255) Or
        (hostmask2 < 0) Or (hostmask2 > 255)) then
      raise EArgumentException.Create('Not a valid subnet mask.');

    // compute the subnet address using bit-wise AND
    sub_net1 := net1 And netmask1;
    sub_net2 := net2 And netmask2;
    sub_host1 := host1 And hostmask1;
    sub_host2 := host2 And hostmask2;

    // display the subnet address
    MaskEdit3.Text :=
      IntToStr(sub_net1) + '.' + IntToStr(sub_net2) + '.' +
      IntToStr(sub_host1) + '.' + IntToStr(sub_host2);
  end;
```

See Also

EditText (see page 1095)

IsMasked (see page 1096)

MaskState (see page 1096)

Text (see page 1097)

ValidateEdit (see page 1091)

ValidateError (see page 1091)

29.1.3.2.22 TMaskEdit.Enabled Property

Controls whether the control responds to mouse, keyboard, and timer events.

Pascal

```
property Enabled;
```

C++

```
__property Enabled;
```

Description

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to False. Some disabled controls appear dimmed (for example: buttons, check boxes, labels), while others (container controls) simply lose their functionality without changing their appearance. If Enabled is set to False, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to True.

Note: This property applies to all TControl descendants.

See Also

Visible

DisableControls

EnableControls

Enabling Mouse

29.1.3.2.23 TMaskEdit.Font Property

Controls the attributes of text written on or in the control.

Pascal

```
property Font;
```

C++

```
__property Font;
```

Description

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

C++ Examples:

```
/*
To use this example, place a TComboBox and TRichEdit or
TMemo on a form.  During form creation, a list of font names
is loaded into the combo box. When the combo box is clicked,
the memo or Rich Edit control font is set to the
corresponding font name in the combo box.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < Screen->Fonts->Count; i++)
   ComboBox1->Items->Add(Screen->Fonts->Strings[i]);
//  ComboBox1->Items = Screen->Fonts;  // Another way to do it.
}

void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
  RichEdit1->Font->Name = ComboBox1->Text;
}
```

Delphi Examples:

```
{
To use this example, place a TComboBox and TRichEdit or
TMemo on a form.  During form creation, a list of font names
is loaded into the combo box. When the combo box is clicked,
the memo or Rich Edit control font is set to the
corresponding font name in the combo box.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  i : Integer;
begin
  for i := 0 to Screen.Fonts.Count - 1 do
   ComboBox1.Items.Add(Screen.Fonts.Strings[i]);
//  ComboBox1.Items := Screen.Fonts;  // Another way to do it.
  RichEdit1.Lines.Text := 'How is it to be you?';
end;
```

```
procedure TForm1.ComboBox1Click(Sender: TObject);
begin
  RichEdit1.Font.Name := ComboBox1.Text;
end;
```

See Also

DesktopFont

ParentFont

Displaying and Editing Text in a Memo Control

29.1.3.2.24 TMaskEdit.ImeMode Property

Determines the behavior of the input method editor (IME).

Pascal

```
property ImeMode;
```

C++

```
__property ImeMode;
```

Description

Set ImeMode to configure the way an IME processes user keystrokes. An IME is a front-end input processor for Asian language characters. The IME hooks all keyboard input, converts it to Asian characters in a conversion window, and sends the converted characters or strings on to the application.

ImeMode allows a control to influence the type of conversion performed by the IME so that it is appropriate for the input expected by the control. For example, a control that only accepts numeric input might specify an ImeMode of imClose, as no conversion is necessary for numeric input.

Note: The value of ImeMode only takes effect when the control receives focus. To change the value of ImeMode when the control already has input focus, call the SetIme method.

See Also

ImeName

ResetIme

ResetImeComposition

SetIme

SetImeCompositionWindow

29.1.3.2.25 TMaskEdit.ImeName Property

Specifies the input method editor (IME) to use for converting keyboard input to Asian language characters.

Pascal

```
property ImeName;
```

C++

```
__property ImeName;
```

Description

Set ImeName to specify which IME to use for converting keystrokes. An IME is a front-end input processor for Asian language

characters. The IME hooks all keyboard input, converts it to Asian characters in a conversion window, and sends the converted characters or strings on to the application.

ImeName must specify one of the IMEs that has been installed through the Windows control panel. The property inspector provides a drop-down list of all currently installed IMEs on the system. At runtime, applications can obtain a list of currently installed IMEs from the global Screen variable.

If ImeName specifies an unavailable IME, the IME that was active when the application started is used instead. No exception is generated.

Note: The value of ImeName only takes effect when the control receives focus. To change the value of ImeName after the control has input focus, call the SetIme method.

See Also

ImeMode

ResetIme

SetIme

DefaultIme

Imes

29.1.3.2.26 TMaskEdit.MaxLength Property

Specifies the maximum number of characters that can appear in the edit control.

Pascal

```
property MaxLength: Integer;
```

C++

```
__property int MaxLength;
```

Description

MaxLength is the length of the EditText. Set MaxLength to limit the number of characters that can appear in the edit control when there is no EditMask. If there is an EditMask, MaxLength is implied by the mask itself, and cannot be changed. The EditText contains blank characters for each character that has not been entered, padding the end or beginning of every optional section, so it remains constant in length.

See Also

EditMask (see page 1092)

EditText (see page 1095)

IsMasked (see page 1096)

Text (see page 1097)

29.1.3.2.27 TMaskEdit.ParentBiDiMode Property

Specifies whether the control uses its parent's BiDiMode.

Pascal

```
property ParentBiDiMode;
```

C++

```
__property ParentBiDiMode;
```

Description

Use ParentBiDiMode to get or set whether the control uses its parent's BiDiMode. When ParentBiDiMode is true, the control's BiDiMode property is always equal to that of its parent. If ParentBiDiMode is false, the control uses its own BiDiMode property.

Set ParentBiDiMode to true for all controls in a form in order to ensure that the entire user interface adjusts to Middle Eastern locales in a uniform manner.

When the value of the control's BiDiMode property changes, ParentBiDiMode becomes false automatically.

See Also

Parent

BiDiMode

29.1.3.2.28 TMaskEdit.ParentColor Property

Determines where a control looks for its color information.

Pascal

```
property ParentColor;
```

C++

```
__property ParentColor;
```

Description

To have a control use the same color as its parent control, set ParentColor to true. If ParentColor is false, the control uses its own Color property.

Set ParentColor to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentColor is true for all controls in a form, changing the background color of the form to gray causes all the controls on the form to also have a gray background.

When the value of a control's Color property changes, ParentColor becomes false automatically.

See Also

Color

Parent

ParentFont

ParentShowHint

29.1.3.2.29 TMaskEdit.ParentCtl3D Property

Determines where a component looks to determine if it should appear three dimensional.

Pascal

```
property ParentCtl3D;
```

C++

```
__property ParentCtl3D;
```

Description

ParentCtl3D is provided for backwards compatibility. It has no effect on 32-bit versions of Windows or NT 4.0 and later.

ParentCtl3D determines whether the control uses its parent's Ctl3D property.

See Also

Ctl3D

29.1.3.2.30 TMaskEdit.ParentDoubleBuffered Property

ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.

Pascal

```
property ParentDoubleBuffered;
```

C++

```
__property ParentDoubleBuffered;
```

Description

ParentDoubleBuffered is a boolean indicating that the DoubleBuffered property of this component is overridden by the value of the parent's DoubleBuffered property.

29.1.3.2.31 TMaskEdit.ParentFont Property

Determines where a control looks for its font information.

Pascal

```
property ParentFont;
```

C++

```
__property ParentFont;
```

Description

To have a control use the same font as its parent control, set ParentFont to true. If ParentFont is false, the control uses its own Font property. Many controls default ParentFont to true so that all the controls in a form or other container present a uniform appearance. When the value of a control's Font property changes, ParentFont becomes false automatically.

When ParentFont is true for a form, the form uses the default font.

See Also

Font

Parent

ParentColor

ParentShowHint

29.1.3.2.32 TMaskEdit.ParentShowHint Property

Determines where a control looks to find out if its Help Hint should be shown.

Pascal

```
property ParentShowHint;
```

C++

```
__property ParentShowHint;
```

Description

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the control uses the ShowHint property value of its parent. If ParentShowHint is false, the control uses the value of its own ShowHint property.

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note: Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See Also

Hint

Parent

ParentColor

ParentFont

ShowHint

29.1.3.2.33 TMaskEdit.PasswordChar Property

Indicates the character, if any, to display in place of the actual characters typed in the control.

Pascal

```
property PasswordChar;
```

C++

```
__property PasswordChar;
```

Description

Use the PasswordChar property to create an edit control that displays a special character in place of any entered text. If PasswordChar is set to the null character (ANSI character zero), the edit control displays its text normally. If PasswordChar is any other character, the edit control displays PasswordChar in place of each character typed. PasswordChar affects the appearance of the edit control only. The value of the Text property reflects the actual characters that are typed.

See Also

Text

29.1.3.2.34 TMaskEdit.PopupMenu Property

Identifies the pop-up menu associated with the control.

Pascal

```
property PopupMenu;
```

C++

```
__property PopupMenu;
```

Description

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the control and clicks the right mouse button. If the TPopupMenu's AutoPopup property is true, the pop-up menu appears automatically. If the menu's AutoPopup property is false, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

C++ Examples:

```
/*
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  PopupMenu1->AutoPopup = true;
  Edit1->PopupMenu = PopupMenu1;
  Edit2->PopupMenu = PopupMenu1;
}

/*
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
*/
void __fastcall TForm1::Copy1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CopyToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Cut1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CutToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Paste1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->PasteFromClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}
/*
```

```
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
    {
        switch (ClickedItem->Tag)
        {
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
            }
        }
    }
}

/*
To edit or destroy an item, grab its pointer via the Items
property.
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}

void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
```

```
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  PopupMenu1.AutoPopup := True;
  Edit1.PopupMenu := PopupMenu1;
  Edit2.PopupMenu := PopupMenu1;
end;

{
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
}
procedure TForm1.Copy1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
   Edit1.CopyToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CopyToClipboard
  else
    Beep;
end;

procedure TForm1.Cut1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.CutToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CutToClipboard
  else
    Beep;
end;

procedure TForm1.Paste1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
```

```
    Edit1.PasteFromClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.PasteFromClipboard
  else
    Beep;
end;
{
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
}

type
  TForm1 = class(TForm)
    AddButton: TButton;
    EditButton: TButton;
    DestroyButton: TButton;
    PopupMenu1: TPopupMenu;
    procedure AddButtonClick(Sender: TObject);
    procedure EditButtonClick(Sender: TObject);
    procedure DestroyButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    procedure PopupMenuItemsClick(Sender: TObject);

  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.AddButtonClick(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
    // The owner (PopupMenu1) will cleanup this menu item.
  NewItem := TMenuItem.Create(PopupMenu1); // create the new item
  index := PopupMenu1.Items.Count;
  PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
  NewItem.Caption := 'Menu Item ' + IntToStr(index);
  NewItem.Tag := index;
  NewItem.OnClick :=
    PopupMenuItemsClick; // assign it an event handler
end;

procedure TForm1.PopupMenuItemsClick(Sender: TObject);
begin
  with Sender as TMenuItem do
  begin
    case Tag of
      0:  ShowMessage('first item clicked');
      1:  ShowMessage('second item clicked');
      2:  ShowMessage('third item clicked');
      3:  ShowMessage('fourth item clicked');
```

```
    end;
  end;
end;

{
To edit or destroy an item, grab its pointer via the Items
property.
}
procedure TForm1.EditButtonClick(Sender: TObject);
var
  ItemToEdit: TMenuItem;
begin
  ItemToEdit := PopupMenu.Items[1];
  ItemToEdit.Caption := 'Changed Caption';
end;

procedure TForm1.DestroyButtonClick(Sender: TObject);
var
  ItemToDelete: TMenuItem;
begin
  ItemToDelete := PopupMenu.Items[2];
  ItemToDelete.Free;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
  for index := 0 to 3 do
  begin
      // The owner (PopupMenu1) will cleanup this menu item.
    NewItem := TMenuItem.Create(PopupMenu1); // create the new item
    PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
    NewItem.Caption := 'Menu Item ' + IntToStr(index);
    NewItem.Tag := index;
    NewItem.OnClick :=
      PopupMenuItemsClick; // assign it an event handler
  end;
end;
```

See Also

AutoPopup (see page 1437)

Popup (see page 1434)

OnContextPopup

29.1.3.2.35 TMaskEdit.ReadOnly Property

Determines whether the user can change the text of the edit control.

Pascal

```
property ReadOnly;
```

C++

```
__property ReadOnly;
```

Description

To restrict the edit control to display only, set the ReadOnly property to true. Set ReadOnly to false to allow the contents of the edit control to be edited.

Setting ReadOnly to true ensures that the text is not altered, while still allowing the user to select text. The selected text can then

be manipulated by the application, or copied to the Clipboard.

See Also

CopyToClipboard

GetSelTextBuf

29.1.3.2.36 TMaskEdit.ShowHint Property

Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.

Pascal

```
property ShowHint;
```

C++

```
__property ShowHint;
```

Description

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either

the control's own ShowHint property must be true, or

the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

C++ Examples:

```
/*
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  Edit1->Hint = "Enter your name";
  Edit1->ShowHint = true;

  char string[10];
  char index[3];
  for(int i = 1; i <= 10; i++)
  {
    strcpy(string, "Item");
    itoa(i, index, 10);
    strcat(string, index);
    ListBox1->Items->Add(string);
  }
  ListBox1->Hint = "Select an item";
  ListBox1->ShowHint = true;
  Application->ShowHint = true;
}
/*
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
*/
```

Delphi Examples:

```
{
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
  Edit1.Hint := 'Enter your name';
  Edit1.ShowHint := True;
  with ListBox1 do
  begin
    for I := 1 to 10 do
      Items.Add('Item ' + IntToStr(I));
    Hint := 'Select an item';
    ShowHint := True;
  end;
  Application.ShowHint := True;
end;
{
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
}
```

See Also

Hint

ParentShowHint

ShowHint

Displaying Fly-over Help

29.1.3.2.37 TMaskEdit.TabOrder Property

Indicates the position of the control in its parent's tab order.

Pascal

```
property TabOrder;
```

C++

```
__property TabOrder;
```

Description

TabOrder is the order in which child windows are visited when the user presses the Tab key. The control with the TabOrder value of 0 is the control that has the focus when the form first appears.

Initially, the tab order is always the order in which the controls were added to the form. The first control added to the form has a TabOrder value of 0, the second is 1, the third is 2, and so on. Change this by changing the TabOrder property.

Each control has a unique tab-order value within its parent. If you change the TabOrder property value of one control to be the same as the value of a second control, the TabOrder value for all the other controls changes. For example, suppose a control is sixth in the tab order. If you change the control's TabOrder property value to 3 (making the control fourth in the tab order), the control that was originally fourth in the tab order now becomes fifth, and the control that was fifth becomes sixth.

Assigning TabOrder a value greater than the number of controls contained in the parent control moves the control to the end of the tab order. The control does not take on the assigned value of TabOrder, but instead is given the number that assures the

control is the last in the tab order.

Note: TabOrder is meaningful only if the TabStop property is true and if the control has a parent. (The TabOrder property of a form is not used unless the form is the child of another form.) A control with a TabOrder of -1 has no parent, and therefore cannot be reached by pressing the Tab key. To remove a parented control from the Tab order, set its TabStop property to false.

See Also

FindNextControl

TabStop

29.1.3.2.38 TMaskEdit.TabStop Property

Determines if the user can tab to a control.

Pascal

```
property TabStop;
```

C++

```
__property TabStop;
```

Description

Use the TabStop to allow or disallow access to the control using the Tab key.

If TabStop is true, the control is in the tab order. If TabStop is false, the control is not in the tab order and the user can't press the Tab key to move to the control.

Note: TabStop is not meaningful for a form unless the form assigns another form to be its parent.

See Also

TabOrder

29.1.3.2.39 TMaskEdit.Text Property

Denotes the underlying text for the masked edit control before the mask has been applied.

Pascal

```
property Text: TMaskedText;
```

C++

```
__property TMaskedText Text;
```

Description

Use Text to determine the underlying value of the text before it has been formatted by the mask. Text may differ from the EditText visible in the masked edit control if the mask specifies that literal characters should be removed, if the mask includes spaces, or if the mask includes characters that have not yet been filled in by the user. Text will not contain the blank characters, spaces will not be replaced by _, and literal characters in the mask will be removed if the mask indicates they should be.

When setting Text, the value of the text is formatted using the mask, and the resulting EditText is used to update the window.

If there is no mask, Text is the text that appears in the edit control.

C++ Examples:

```
/*
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
```

```
  to contain the empty string, to use the Courier font and to
  use a particular mask, corresponding to IP address coding convetions.
  The mask edit boxes display the IP address, the subnet mask
  and the subnet address correspondingly. When the user exits the
  subnet mask edit box, the subnet address is automatically calculated
  using bit-wise AND operations and displayed in the third mask edit box.
  An additional test is made to assure that the values in the IP address
  and in the subnet mask are in the interval from 0 to 255 ,
  as this validation test cannot be accomplished
  using the edit mask property alone.
  */
  void __fastcall TForm1::FormCreate(TObject *Sender)
  {
    // initialize the maskedit boxes to contain the empty string
    MaskEdit1->Text = "";
    MaskEdit2->Text = "";
    MaskEdit3->Text = "";

    // use the Courier font in the maskedit boxes
    MaskEdit1->Font->Name = "Courier";
    MaskEdit2->Font->Name = "Courier";
    MaskEdit3->Font->Name = "Courier";

    // initialize the mask of each maskedit box
    // according to IP address coding conventions
    MaskEdit1->EditMask = "!099.099.099.099;1; ";
    MaskEdit2->EditMask = "!099.099.099.099;1; ";
    MaskEdit3->EditMask = "!099.099.099.099;1; ";

    MaskEdit2->OnExit = MaskEdit2Exit;
  }

  void __fastcall TForm1::MaskEdit2Exit(TObject *Sender)
  {
    // extract the net and host address from the IP
    String IP = MaskEdit1->Text;
    int net1 = IP.SubString(0, 3).TrimRight().ToInt();
    int net2 = IP.SubString(5, 3).TrimRight().ToInt();
    int host1 = IP.SubString(9, 3).TrimRight().ToInt();
    int host2 = IP.SubString(13, 3).TrimRight().ToInt();

    // a range test that you cannot validate through edit masks
    if (net1 < 0 || net1 > 255 ||
        net2 < 0 || net2 > 255 ||
        host1 < 0 || host1 > 255 ||
        host2 < 0 || host2 > 255)
      throw(Exception("Not a valid IP address."));

    // extract the net and host mask from the subnet mask
    String mask = MaskEdit2->Text;
    int netmask1 = mask.SubString(0, 3).TrimRight().ToInt();
    int netmask2 = mask.SubString(5, 3).TrimRight().ToInt();
    int hostmask1 = mask.SubString(9, 3).TrimRight().ToInt();
    int hostmask2 = mask.SubString(13, 3).TrimRight().ToInt();

    // a range test that you cannot validate through edit masks
    if (netmask1 < 0 || netmask1 > 255 ||
        netmask2 < 0 || netmask2 > 255 ||
        hostmask1 < 0 || hostmask1 > 255 ||
        hostmask2 < 0 || hostmask2 > 255)
      throw(Exception("Not a valid subnet mask."));

    // compute the subnet address using bit-wise AND
    int sub_net1 = net1 & netmask1;
    int sub_net2 = net2 & netmask2;
    int sub_host1 = host1 & hostmask1;
```

```
  int sub_host2 = host2 & hostmask2;

  // display the subnet address
  MaskEdit3->Text =
    String(sub_net1) + "." + String(sub_net2) + "." +
    String(sub_host1) + "." + String(sub_host2);
}
```

Delphi Examples:

```
{
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255,
as this validation test cannot be accomplished
using the edit mask property alone.
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  { initialize the maskedit boxes to contain the empty string }
  MaskEdit1.Text := '';
  MaskEdit2.Text := '';
  MaskEdit3.Text := '';

  { use the Courier font in the maskedit boxes }
  MaskEdit1.Font.Name := 'Courier';
  MaskEdit2.Font.Name := 'Courier';
  MaskEdit3.Font.Name := 'Courier';

  {
  initialize the mask of each maskedit box
  according to IP address coding conventions
  }
  MaskEdit1.EditMask := '!099.099.099.099;1; ';
  MaskEdit2.EditMask := '!099.099.099.099;1; ';
  MaskEdit3.EditMask := '!099.099.099.099;1; ';

  MaskEdit2.OnExit := MaskEdit2Exit;
end;

procedure TForm1.MaskEdit2Exit(Sender: TObject);
var
  net1, net2, host1, host2,
  netmask1, netmask2, hostmask1, hostmask2,
  sub_net1, sub_net2, sub_host1, sub_host2: Integer;

  IP, mask: String;

begin
  // extract the net and host address from the IP
  IP := MaskEdit1.Text;
  net1 := StrToInt(TrimRight(Copy(IP, 0, 3)));
  net2 := StrToInt(TrimRight(Copy(IP, 5, 3)));
  host1 := StrToInt(TrimRight(Copy(IP, 9, 3)));
  host2 := StrToInt(TrimRight(Copy(IP, 13, 3)));

  // a range test that you cannot validate through edit masks
  if ((net1 < 0) Or (net1 > 255) Or
      (net2 < 0) Or (net2 > 255) Or
```

```
        (host1 < 0) Or (host1 > 255) Or
        (host2 < 0) Or (host2 > 255)) then
      raise EArgumentException.Create('Not a valid IP address.');

    // extract the net and host mask from the subnet mask
    mask := MaskEdit2.Text;
    netmask1 := StrToInt(TrimRight(Copy(mask, 0, 3)));
    netmask2 := StrToInt(TrimRight(Copy(mask, 5, 3)));
    hostmask1 := StrToInt(TrimRight(Copy(mask, 9, 3)));
    hostmask2 := StrToInt(TrimRight(Copy(mask, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((netmask1 < 0) Or (netmask1 > 255) Or
        (netmask2 < 0) Or (netmask2 > 255) Or
        (hostmask1 < 0) Or (hostmask1 > 255) Or
        (hostmask2 < 0) Or (hostmask2 > 255)) then
      raise EArgumentException.Create('Not a valid subnet mask.');

    // compute the subnet address using bit-wise AND
    sub_net1 := net1 And netmask1;
    sub_net2 := net2 And netmask2;
    sub_host1 := host1 And hostmask1;
    sub_host2 := host2 And hostmask2;

    // display the subnet address
    MaskEdit3.Text :=
      IntToStr(sub_net1) + '.' + IntToStr(sub_net2) + '.' +
      IntToStr(sub_host1) + '.' + IntToStr(sub_host2);
  end;
```

See Also

EditMask (see page 1092)

EditText (see page 1095)

GetTextLen (see page 1087)

IsMasked (see page 1096)

ReformatText (see page 1089)

29.1.3.2.40 TMaskEdit.TextHint Property

A hint or message to be displayed when the Text property is empty.

Pascal

```
property TextHint;
```

C++

```
__property TextHint;
```

Description

Use TextHint to cue the user on what kind of entry is expected in the text field. Runtime themes must be enabled.

29.1.3.2.41 TMaskEdit.Visible Property

Determines whether the component appears onscreen.

Pascal

```
property Visible;
```

C++

```
__property Visible;
```

Description

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

Calling the Show method sets the control's Visible property to true. Calling the Hide method sets it to false.

C++ Examples:

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
    pPage->PageControl = PageControl1;
    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this); // The owner will clean this up.
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
```

```
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
    begin
      PageControl := PageControl1;
      Caption := 'TabSheet' + IntToStr(i);
      with TEdit.Create(Self) do
      begin
        Parent := PageControl1.Pages[i];
        Left := Random(PageControl1.ActivePage.ClientWidth - Width);
        Top := Random(PageControl1.ActivePage.ClientHeight - Height);
      end;
    end;
    PageControl1Change(Sender);
end;
```

See Also

Enabled

Hide

Show

VisibleChanging

29.1.3.3 TMaskEdit Events

The events of the TMaskEdit class are listed here.

Published Events

	Name	Description
	OnChange (see page 1142)	Occurs when the text for the edit control may have changed.
	OnClick (see page 1142)	Occurs when the user clicks the control.
	OnDblClick (see page 1144)	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.
	OnDragDrop (see page 1144)	Occurs when the user drops an object being dragged.
	OnDragOver (see page 1147)	Occurs when the user drags an object over a control.
	OnEndDock (see page 1150)	Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.
	OnEndDrag (see page 1151)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.
	OnEnter (see page 1152)	Occurs when a control receives the input focus.
	OnExit (see page 1154)	Occurs when the input focus shifts away from one control to another.
	OnKeyDown (see page 1158)	Occurs when a user presses any key while the control has focus.
	OnKeyPress (see page 1162)	Occurs when a key is pressed.
	OnKeyUp (see page 1163)	Occurs when the user releases a key that has been pressed.
	OnMouseActivate (see page 1165)	Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.
	OnMouseDown (see page 1166)	Occurs when the user presses a mouse button with the mouse pointer over a control.
	OnMouseEnter (see page 1168)	Occurs when the user moves the mouse into a control.
	OnMouseLeave (see page 1168)	Occurs when the user moves the mouse outside of a control.
	OnMouseMove (see page 1168)	Occurs when the user moves the mouse pointer while the mouse pointer is over a control.
	OnMouseUp (see page 1171)	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.
	OnStartDock (see page 1173)	Occurs when the user begins to drag a control with a DragKind of dkDock.
	OnStartDrag (see page 1173)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

29.1.3.3.1 TMaskEdit.OnChange Event

Occurs when the text for the edit control may have changed.

Pascal

```
property OnChange;
```

C++

```
__property OnChange;
```

Description

Write an OnChange event handler to take specific action whenever the text for the edit control may have changed. Use the Modified property to see if a change actually occurred. The Text property of the edit control will already be updated to reflect any changes. This event provides the first opportunity to respond to modifications that the user types into the edit control.

See Also

Change

Text

29.1.3.3.2 TMaskEdit.OnClick Event

Occurs when the user clicks the control.

Pascal

```
property OnClick;
```

C++

```
__property OnClick;
```

Description

Use the OnClick event handler to respond when the user clicks the control. If the control has an associated action, and that action has an OnExecute method, the action's OnExecute method responds to click events unless it is superseded by an OnClick event handler.

Usually OnClick occurs when the user presses and releases the left mouse button with the mouse pointer over the control. This event can also occur when:

- The user selects an item in a grid, outline, list, or combo box by pressing an arrow key.
- The user presses Spacebar while a button or check box has focus.
- The user presses Enter when the active form has a default button (specified by the Default property).
- The user presses Esc when the active form has a cancel button (specified by the Cancel property).
- The user presses the accelerator key for a button or check box. For example, if the value of the Caption property of a check box is '&Bold', the B is underlined at runtime and the OnClick event of the check box is triggered when the user presses Alt+B. However, focus does not move to the control in these instances.
- The Checked property of a radio button is set to true.
- The value of the Checked property of a check box is changed.
- The Click method of a menu item is called.

For a form, an OnClick event occurs when the user clicks a blank area of the form or on a disabled component.

C++ Examples:

```
/*
```

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
    pPage->PageControl = PageControl1;
    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this); // The owner will clean this up.
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
    begin
      PageControl := PageControl1;
      Caption := 'TabSheet' + IntToStr(i);
      with TEdit.Create(Self) do
      begin
        Parent := PageControl1.Pages[i];
        Left := Random(PageControl1.ActivePage.ClientWidth - Width);
        Top := Random(PageControl1.ActivePage.ClientHeight - Height);
      end;
    end;
    PageControl1Change(Sender);
```

```
end;
```

See Also

OnDblClick

Cancel

Default

Click (see page 1375)

OnExecute

Action

Adding Silent Video Clips to an Application

29.1.3.3.3 TMaskEdit.OnDblClick Event

Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.

Pascal

```
property OnDblClick;
```

C++

```
__property OnDblClick;
```

Description

Use the OnDblClick event to respond to mouse double-clicks.

See Also

OnClick

Responding to User Actions at Runtime

29.1.3.3.4 TMaskEdit.OnDragDrop Event

Occurs when the user drops an object being dragged.

Pascal

```
property OnDragDrop;
```

C++

```
__property OnDragDrop;
```

Description

Use the OnDragDrop event handler to specify what happens when the user drops an object. The Source parameter of the OnDragDrop event is the object being dropped, and the Sender is the control the object is being dropped on. The X and Y parameters are the coordinates of the mouse positioned over the control.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
```

```
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}

void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
```

```
  onto the panel:
  Note: Set the label's DragMode property to dmAutomatic.
  }
  type
    TForm1 = class(TForm)
      Panel1: TPanel;
      Label1: TLabel;
      procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
        State: TDragState; var Accept: Boolean);
      procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
        State: TDragState; var Accept: Boolean);
      procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
    private
      { Private declarations }
    public
      { Public declarations }
    end;

  var
    Form1: TForm1;

  implementation

  {$R *.dfm}

  procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  begin
    TForm1(Source).Left := X;
    TForm1(Source).Top := Y;
  end;

  procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
    State: TDragState; var Accept: Boolean);
  begin
    Accept := Source is TLabel;
  end;

  procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
    State: TDragState; var Accept: Boolean);
  begin
    Accept := False;
    if (Source is TLabel) and (State = dsDragEnter) then
      (Source as TLabel).EndDrag(False);
  end;
  {
  This code requires a list box and three labels, each with a
  different font and color.  The DragMode property for each of
  the labels is dmAutomatic.  The user can select a label and
  drag it to a list box and drop it. When the label is dropped,
  the items in the list box assume the color and font of the
  dropped label.
  }

  procedure TForm1.FormCreate(Sender: TObject);
  begin
    ListBox1.Items.Add('Not');
    ListBox1.Items.Add('In');
    ListBox1.Items.Add('Alphabetical');
    ListBox1.Items.Add('Order');
  end;

  // This OnDragOver event handler permits the list box to
  // accept a dropped label:

  procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
    State: TDragState; var Accept: Boolean);
```

```
begin
  Accept := Source is TLabel;
end;

// This OnDragDrop event handler implements the drop behavior.

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
    with Sender as TListBox do
    begin
      Font := (Source as TLabel).Font;
      Color := (Source as TLabel).Color;
      DoubleBuffered := true;
      Color := clWindow;
    end;
  end;
end;
```

See Also

BeginDrag

DragDrop

Dragging

EndDrag

OnDragOver

OnEndDrag

OnStartDrag

TDragObject

29.1.3.3.5 TMaskEdit.OnDragOver Event

Occurs when the user drags an object over a control.

Pascal

```
property OnDragOver;
```

C++

```
__property OnDragOver;
```

Description

Use an OnDragOver event to signal that the control can accept a dragged object so the user can drop or dock it.

Within the OnDragOver event handler, change the Accept parameter to false to reject the dragged object. Leave Accept as true to allow the user to drop or dock the dragged object on the control.

To change the shape of the cursor, indicating that the control can accept the dragged object, change the value of the DragCursor property for the control before the OnDragOver event occurs.

The Source is the object being dragged, the Sender is the potential drop or dock site, and X and Y are screen coordinates in pixels. The State parameter specifies how the dragged object is moving over the control.

Note: Within the OnDragOver event handler, the Accept parameter defaults to true. However, if an OnDragOver event handler is not supplied, the control rejects the dragged object, as if the Accept parameter were changed to false.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}

void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
```

```
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
}
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  TForm1(Source).Left := X;
  TForm1(Source).Top := Y;
end;

procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := False;
  if (Source is TLabel) and (State = dsDragEnter) then
    (Source as TLabel).EndDrag(False);
end;
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
dropped label.
}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Not');
  ListBox1.Items.Add('In');
```

```
    ListBox1.Items.Add('Alphabetical');
    ListBox1.Items.Add('Order');
  end;

  // This OnDragOver event handler permits the list box to
  // accept a dropped label:

  procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
    State: TDragState; var Accept: Boolean);
  begin
    Accept := Source is TLabel;
  end;

  // This OnDragDrop event handler implements the drop behavior.

  procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
  begin
    if (Sender is TListBox) and (Source is TLabel) then
    begin
      with Sender as TListBox do
      begin
        Font := (Source as TLabel).Font;
        Color := (Source as TLabel).Color;
        DoubleBuffered := true;
        Color := clWindow;
      end;
    end;
  end;
```

See Also

DragCursor

OnDragDrop

OnEndDrag

OnStartDrag

TDragObject

DragKind

OnEndDock

OnDockDrop

OnStartDock

29.1.3.3.6 TMaskEdit.OnEndDock Event

Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.

Pascal

```
property OnEndDock;
```

C++

```
__property OnEndDock;
```

Description

Use OnEndDock to specify actions or special processing that when a drag-and-dock operation stops.

See Also

DragKind

OnDragOver

OnDockDrop

OnStartDock

29.1.3.3.7 TMaskEdit.OnEndDrag Event

Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.

Pascal

```
property OnEndDrag;
```

C++

```
__property OnEndDrag;
```

Description

Use the OnEndDrag event handler to specify any special processing that occurs when dragging stops.

C++ Examples:

```
/*
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
*/
void __fastcall TForm1::Label1EndDrag(TObject *Sender, TObject *Target, int X,
      int Y)
{
  if (Sender->ClassNameIs("TLabel"))
  {
    AnsiString S = (dynamic_cast<TLabel *>(Sender))->Name + " was dropped and ";
    if (Target)
      S = S + "accepted!";
    else
      S = S + "rejected.";
    Status->Caption = S;
  }
}
```

Delphi Examples:

```
{
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
}
procedure TForm1.Label1EndDrag(Sender, Target: TObject; X, Y: Integer);
var
  S: string;
begin
  S := (Sender as TLabel).Name + ' was dropped... and ';
  if Target <> nil then
    S := S + 'accepted!'
  else
    S := S + 'rejected!';
  Status.Caption := S;
end;
```

See Also

OnDragDrop

OnDragOver

OnStartDrag

DragKind

29.1.3.3.8 TMaskEdit.OnEnter Event

Occurs when a control receives the input focus.

Pascal

```
property OnEnter;
```

C++

```
__property OnEnter;
```

Description

Use the OnEnter event handler to cause any special processing to occur when a control becomes active.

The OnEnter event does not occur when switching between forms or between another application and the application that includes the control.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnEnter event occurs for the container before the OnEnter event of the contained control.

Similarly, an OnExit event of the container occurs after the OnExit event of the control in a container when focus moves to another control outside the container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

C++ Examples:

```
/*
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
*/
TWinControl *myActiveControl;

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  FontDialog1->Options << fdApplyButton;
  FontDialog1->Execute();
}

void __fastcall TForm1::FontDialog1Apply(TObject *Sender, HWND Wnd)
{
  if (myActiveControl->ClassNameIs("TEdit"))
   (dynamic_cast<TEdit *>(myActiveControl))->Font->Assign(FontDialog1->Font);
  else if (myActiveControl->ClassNameIs("TRichEdit"))
    (dynamic_cast<TRichEdit *>(myActiveControl))->SelAttributes->Assign(FontDialog1->Font);
  else
    MessageBeep(0);
```

```
}

void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  myActiveControl = ActiveControl;
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
   FontDialog1.Options := FontDialog1.Options + [fdApplyButton];
   FontDialog1.Execute;
end;

procedure TForm1.Edit1Enter(Sender: TObject);
begin
  myActiveControl := ActiveControl;
end;

procedure TForm1.FontDialog1Apply(Sender: TObject; Wnd: HWND);
begin
  if myActiveControl is TEdit then
    with myActiveControl as TEdit do
       Font.Assign(TFontDialog(Sender).Font)
  else if myActiveControl is TRichEdit then
    with myActiveControl as TRichEdit do
      SelAttributes.Assign(TFontDialog(Sender).Font)
  else
    Beep;
```

```
end;
{
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
}
procedure TForm1.Edit1Enter(Sender: TObject);
begin
  Edit1.Color := clYellow;
end;

procedure TForm1.Edit1Exit(Sender: TObject);
begin
  Edit1.Color := clWindow;
end;

procedure TForm1.Memo1Enter(Sender: TObject);
begin
  Memo1.Color := clYellow;
end;

procedure TForm1.Memo1Exit(Sender: TObject);
begin
  Memo1.Color := clWindow;
end;
```

See Also

OnExit

ActiveControl

OnActivate

Using a Single Navigator for Multiple Datasets

29.1.3.3.9 TMaskEdit.OnExit Event

Occurs when the input focus shifts away from one control to another.

Pascal

```
property OnExit;
```

C++

```
__property OnExit;
```

Description

Use the OnExit event handler to provide special processing when the control ceases to be active.

The OnExit event does not occur when switching between forms or between another application and your application.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnExit event occurs for the control inside the container before the OnExit event of the container.

Similarly, an OnEnter event of the container occurs before the OnEnter event of the control in a container when focus moves to a control inside a container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK

button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

Note: In some control classes the ActiveControl property updates before the OnExit event occurs.

C++ Examples:

```
/*
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255 ,
as this validation test cannot be accomplished
using the edit mask property alone.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  // initialize the maskedit boxes to contain the empty string
  MaskEdit1->Text = "";
  MaskEdit2->Text = "";
  MaskEdit3->Text = "";

  // use the Courier font in the maskedit boxes
  MaskEdit1->Font->Name = "Courier";
  MaskEdit2->Font->Name = "Courier";
  MaskEdit3->Font->Name = "Courier";

  // initialize the mask of each maskedit box
  // according to IP address coding conventions
  MaskEdit1->EditMask = "!099.099.099.099;1; ";
  MaskEdit2->EditMask = "!099.099.099.099;1; ";
  MaskEdit3->EditMask = "!099.099.099.099;1; ";

  MaskEdit2->OnExit = MaskEdit2Exit;
}

void __fastcall TForm1::MaskEdit2Exit(TObject *Sender)
{
  // extract the net and host address from the IP
  String IP = MaskEdit1->Text;
  int net1 = IP.SubString(0, 3).TrimRight().ToInt();
  int net2 = IP.SubString(5, 3).TrimRight().ToInt();
  int host1 = IP.SubString(9, 3).TrimRight().ToInt();
  int host2 = IP.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (net1 < 0 || net1 > 255 ||
      net2 < 0 || net2 > 255 ||
      host1 < 0 || host1 > 255 ||
      host2 < 0 || host2 > 255)
    throw(Exception("Not a valid IP address."));

  // extract the net and host mask from the subnet mask
  String mask = MaskEdit2->Text;
  int netmask1 = mask.SubString(0, 3).TrimRight().ToInt();
  int netmask2 = mask.SubString(5, 3).TrimRight().ToInt();
  int hostmask1 = mask.SubString(9, 3).TrimRight().ToInt();
```

```
    int hostmask2 = mask.SubString(13, 3).TrimRight().ToInt();

    // a range test that you cannot validate through edit masks
    if (netmask1 < 0 || netmask1 > 255 ||
        netmask2 < 0 || netmask2 > 255 ||
        hostmask1 < 0 || hostmask1 > 255 ||
        hostmask2 < 0 || hostmask2 > 255)
      throw(Exception("Not a valid subnet mask."));

    // compute the subnet address using bit-wise AND
    int sub_net1 = net1 & netmask1;
    int sub_net2 = net2 & netmask2;
    int sub_host1 = host1 & hostmask1;
    int sub_host2 = host2 & hostmask2;

    // display the subnet address
    MaskEdit3->Text =
      String(sub_net1) + "." + String(sub_net2) + "." +
      String(sub_host1) + "." + String(sub_host2);
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255,
as this validation test cannot be accomplished
using the edit mask property alone.
}
procedure TForm1.FormCreate(Sender: TObject);
begin
```

```
    { initialize the maskedit boxes to contain the empty string }
    MaskEdit1.Text := '';
    MaskEdit2.Text := '';
    MaskEdit3.Text := '';

    { use the Courier font in the maskedit boxes }
    MaskEdit1.Font.Name := 'Courier';
    MaskEdit2.Font.Name := 'Courier';
    MaskEdit3.Font.Name := 'Courier';

    {
    initialize the mask of each maskedit box
    according to IP address coding conventions
    }
    MaskEdit1.EditMask := '!099.099.099.099;1; ';
    MaskEdit2.EditMask := '!099.099.099.099;1; ';
    MaskEdit3.EditMask := '!099.099.099.099;1; ';

    MaskEdit2.OnExit := MaskEdit2Exit;
  end;

  procedure TForm1.MaskEdit2Exit(Sender: TObject);
  var
    net1, net2, host1, host2,
    netmask1, netmask2, hostmask1, hostmask2,
    sub_net1, sub_net2, sub_host1, sub_host2: Integer;

    IP, mask: String;

  begin
    // extract the net and host address from the IP
    IP := MaskEdit1.Text;
    net1 := StrToInt(TrimRight(Copy(IP, 0, 3)));
    net2 := StrToInt(TrimRight(Copy(IP, 5, 3)));
    host1 := StrToInt(TrimRight(Copy(IP, 9, 3)));
    host2 := StrToInt(TrimRight(Copy(IP, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((net1 < 0) Or (net1 > 255) Or
        (net2 < 0) Or (net2 > 255) Or
        (host1 < 0) Or (host1 > 255) Or
        (host2 < 0) Or (host2 > 255)) then
      raise EArgumentException.Create('Not a valid IP address.');

    // extract the net and host mask from the subnet mask
    mask := MaskEdit2.Text;
    netmask1 := StrToInt(TrimRight(Copy(mask, 0, 3)));
    netmask2 := StrToInt(TrimRight(Copy(mask, 5, 3)));
    hostmask1 := StrToInt(TrimRight(Copy(mask, 9, 3)));
    hostmask2 := StrToInt(TrimRight(Copy(mask, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((netmask1 < 0) Or (netmask1 > 255) Or
        (netmask2 < 0) Or (netmask2 > 255) Or
        (hostmask1 < 0) Or (hostmask1 > 255) Or
        (hostmask2 < 0) Or (hostmask2 > 255)) then
      raise EArgumentException.Create('Not a valid subnet mask.');

    // compute the subnet address using bit-wise AND
    sub_net1 := net1 And netmask1;
    sub_net2 := net2 And netmask2;
    sub_host1 := host1 And hostmask1;
    sub_host2 := host2 And hostmask2;

    // display the subnet address
    MaskEdit3.Text :=
```

```
      IntToStr(sub_net1) + '.' + IntToStr(sub_net2) + '.' +
      IntToStr(sub_host1) + '.' + IntToStr(sub_host2);
  end;
  {
  This example uses an edit box and a memo control on a form.
  When either the edit box or the memo is the active control,
  it is colored yellow. When the active control becomes
  inactive, the color of the control returns to the Windows
  system color for a window.  These event handlers could also
  be shared.
  }
  procedure TForm1.Edit1Enter(Sender: TObject);
  begin
    Edit1.Color := clYellow;
  end;

  procedure TForm1.Edit1Exit(Sender: TObject);
  begin
    Edit1.Color := clWindow;
  end;

  procedure TForm1.Memo1Enter(Sender: TObject);
  begin
    Memo1.Color := clYellow;
  end;

  procedure TForm1.Memo1Exit(Sender: TObject);
  begin
    Memo1.Color := clWindow;
  end;
```

See Also

OnEnter

ActiveControl

29.1.3.3.10 TMaskEdit.OnKeyDown Event

Occurs when a user presses any key while the control has focus.

Pascal

```
property OnKeyDown;
```

C++

```
__property OnKeyDown;
```

Description

Use the OnKeyDown event handler to specify special processing to occur when a key is pressed. The OnKeyDown handler can respond to keyboard keys, including function keys and keys combined with the `Shift`, `Alt`, and `Ctrl` keys, and pressed mouse buttons.

Note: TCheckBox does not fire OnKeyDown events for arrow keys.

The TKeyEvent type points to a method that handles keyboard events.

The Key parameter is the key on the keyboard. For non-alphanumeric keys, use virtual key codes to determine the key pressed. For more information, see Virtual Key codes.

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

An application gets Windows WM_KEYDOWN messages for all keys when the user presses a key. These messages indirectly fire the OnKeyDown event. If you set the Key to 0 there you prevent only any further processing of this message. But for keys

that generate characters Windows also produces WM_CHAR. At the time your OnKeyDown event fires, the WM_CHAR message for the key will already be in the message queue. Setting Key to 0 does not stop it from being delivered, so it fires the OnKeyPress event. You must set the Key to #0 to prevent the control from intercepting the WM_CHAR message.

This method of organizing key processing has advantages. Code that only deals with characters, including control characters like #13 for carriage return, #3 for Ctrl-C and so on, should go into the OnKeyPress event. Code that deals with keys that do not generate characters should be put into the OnKeyDown event.

C++ Examples:

```
/*
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
*/
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  if (Key == VK_F1)
  {
    FormColor = Form1->Color;
    Form1->Color = clAqua;
    if (ComboBox1->Text == "KeyDown")
      ComboBox1->Text = "KeyDown2";
    else
      ComboBox1->Text = "KeyDown";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}

void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key, TShiftState Shift)
{
  if (Key == VK_F1)
  {
    Form1->Color = FormColor;
    if (ComboBox1->Text == "KeyUp")
      ComboBox1->Text = "KeyUp2";
    else
      ComboBox1->Text = "KeyUp";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}
/*
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
*/
#include <Printers.hpp>

void __fastcall TForm1::FormCreate(TObject *Sender)
{
   KeyPreview = True;

}

void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)

{
  if (Key == VK_ESCAPE && Printer()->Printing)
```

```
    {
      Printer()->Abort();
      MessageDlg("Printing aborted", mtInformation,
                 TMsgDlgButtons() << mbOK,0);
    }
  }

  void __fastcall TForm1::Button1Click(TObject *Sender)
  {
    TMemo *Memo1 = new TMemo(Form1); // The parent (Form1) will clean up this object.
    Memo1->Parent = Form1;
    Memo1->Visible = True;
    Memo1->Width = 700;
    if (OpenDialog1->Execute())
    {
      Memo1->Lines->LoadFromFile(OpenDialog1->FileName);

      Printer()->BeginDoc();
      int X = 200;
      int Y = 200;
      for (int I = 0; I <= 10; I++)
        if (!Printer()->Aborted)
        {
          Printer()->Canvas->TextOut(X, Y, Memo1->Lines->Strings[I]);
          Y = Y + 80;
          if (Y > (Printer()->PageHeight - 300))
          {
            Y = 200;
            Printer()->NewPage();
            Sleep(1000);  // to give you time to abort!
          }
        }
      if (!Printer()->Aborted) Printer()->EndDoc();
    }
  }
```

Delphi Examples:

```
{
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
}
var
  FormColor: TColor;
procedure TForm1.FormCreate(Sender: TObject);
begin
  KeyPreview := True;
  FormColor := Form1.Color;
end;

procedure TForm1.FormKeyDown(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := clAqua;
    if (ComboBox1.Text = 'KeyDown') then
      ComboBox1.Text := 'KeyDown2'
    else
      ComboBox1.Text := 'KeyDown';
    WriteLn(ComboBox1.Text);
  end;
```

```
end;

procedure TForm1.FormKeyUp(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := FormColor;
    if (ComboBox1.Text = 'KeyUp') then
      ComboBox1.Text := 'KeyUp2'
    else
      ComboBox1.Text := 'KeyUp';
    WriteLn(ComboBox1.Text);
  end;
end;
{
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
}

uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  I, X, Y: Integer;
  Memo1 : TMemo;
  r: TRect;
begin
  Memo1 := TMemo.Create(Form1);
  Memo1.Parent := Form1;
  Memo1.Visible := True;
  Memo1.Width := 700;
  if (OpenDialog1.Execute) then
  begin
    Memo1.Lines.LoadFromFile(OpenDialog1.FileName);

    Printer.BeginDoc;
    X := 200;
    Y := 200;
    for I := 0 to 140 do
      if (not Printer.Aborted) then
      begin
        Printer.Canvas.TextOut(X, Y, Memo1.Lines[I]);
        Y := Y + 80;
        if (Y > (Printer.PageHeight - 300)) then
        begin
          Y := 200;
          Printer.NewPage;
          Sleep(1000);  // to give you time to abort!
        end;
      end;
    if (not Printer.Aborted) then Printer.EndDoc;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
   KeyPreview := True;
end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
if (Key=VK_ESCAPE) and Printer.Printing then
  begin
  Printer.Abort;
```

```
    MessageDlg('Printing aborted', mtInformation, [mbOK],0);
    end;
  end;
```

See Also

KeyPreview

OnKeyPress

OnKeyUp

29.1.3.3.11 TMaskEdit.OnKeyPress Event

Occurs when a key is pressed.

Pascal

```
property OnKeyPress;
```

C++

```
__property OnKeyPress;
```

Description

Use the OnKeyPress event handler to make something happen as a result of a single character key press.

The Key parameter in the OnKeyPress event handler is of type Char; therefore, the OnKeyPress event registers the ASCII character of the key pressed. Keys that don't correspond to an ASCII Char value (Shift or F1, for example) don't generate an OnKeyPress event. Key combinations (such as Shift+A), generate only one OnKeyPress event (for this example, Shift+A results in a Key value of "A" if Caps Lock is off). To respond to non-ASCII keys or key combinations, use the OnKeyDown or OnKeyUp event handlers.

An application gets Windows WM_KEYDOWN messages for all keys when the user presses a key. These messages indirectly fire the OnKeyDown event. If you set the Key to 0 there you prevent only any further processing of this message. But for keys that generate characters Windows also produces WM_CHAR. At the time your OnKeyDown event fires, the WM_CHAR message for the key will already be in the message queue. Setting Key to 0 does not stop it from being delivered, so it fires the OnKeyPress event. You must set the Key to #0 to prevent the control from intercepting the WM_CHAR message.

This method of organizing key processing has advantages. Code that only deals with characters, including control characters like #13 for carriage return, #3 for Ctrl-C and so on, should go into the OnKeyPress event. Code that deals with keys that do not generate characters should be put into the OnKeyDown event.

C++ Examples:

```
/*
This event handler displays a message dialog box specifying
which key was pressed:
*/

void __fastcall TForm1::FormKeyPress(TObject *Sender, char &Key)
{
  char keyString[25];
  keyString[0] = Key;
  strcpy(&keyString[1], " Was Pressed");
  TMsgDlgButtons() << mbOK;
  MessageDlg(AnsiString(keyString), TMsgDlgType(mtInformation), TMsgDlgButtons() << mbOK, 0);
}
```

Delphi Examples:

```
{
This event handler displays a message dialog box specifying
which key was pressed:
```

```
}
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  MessageDlg(Key + ' has been pressed', mtInformation, [mbOK], 0)
end;
```

See Also

OnKeyDown

OnKeyUp

KeyPreview

29.1.3.3.12 TMaskEdit.OnKeyUp Event

Occurs when the user releases a key that has been pressed.

Pascal

```
property OnKeyUp;
```

C++

```
__property OnKeyUp;
```

Description

Use the OnKeyUp event handler to provide special processing that occurs when a key is released. The OnKeyUp handler can respond to all keyboard keys, including function keys and keys combined with the Shift, Alt, and Ctrl keys.

The TKeyEvent type points to a method that handles keyboard events. The Key parameter is the key on the keyboard. For non-alphanumeric keys, you must use virtual key codes to determine the key pressed. For more information, see Virtual Key codes.

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

C++ Examples:

```
/*
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
*/
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  if (Key == VK_F1)
  {
    FormColor = Form1->Color;
    Form1->Color = clAqua;
    if (ComboBox1->Text == "KeyDown")
      ComboBox1->Text = "KeyDown2";
    else
      ComboBox1->Text = "KeyDown";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}

void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key, TShiftState Shift)
{
```

```
  if (Key == VK_F1)
  {
    Form1->Color = FormColor;
    if (ComboBox1->Text == "KeyUp")
      ComboBox1->Text = "KeyUp2";
    else
      ComboBox1->Text = "KeyUp";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}
```

Delphi Examples:

```
{
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
}
var
  FormColor: TColor;
procedure TForm1.FormCreate(Sender: TObject);
begin
  KeyPreview := True;
  FormColor := Form1.Color;
end;

procedure TForm1.FormKeyDown(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := clAqua;
    if (ComboBox1.Text = 'KeyDown') then
      ComboBox1.Text := 'KeyDown2'
    else
      ComboBox1.Text := 'KeyDown';
    WriteLn(ComboBox1.Text);
  end;
end;

procedure TForm1.FormKeyUp(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := FormColor;
    if (ComboBox1.Text = 'KeyUp') then
      ComboBox1.Text := 'KeyUp2'
    else
      ComboBox1.Text := 'KeyUp';
    WriteLn(ComboBox1.Text);
  end;
end;
```

See Also

OnKeyDown

OnKeyPress

29.1.3.3.13 TMaskEdit.OnMouseActivate Event

Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.

Pascal

```
property OnMouseActivate;
```

C++

```
__property OnMouseActivate;
```

Description

Use the OnMouseActivate event handler to implement any special processing that should occur as a result of pressing a mouse button on a control when the parent top-level form is not active.

The event type is:

```
TMouseActivateEvent = procedure (Sender: TObject; Button: TMouseButton; ShiftState:
TShiftState;
X, Y: Integer; HitTest: Integer; var MouseActivate: TMouseActivate) of object;
```

When you click on a control and the parent top-level form is not active, this event fires on the control that the mouse cursor is over. MouseActivate is a protected virtual function in TControl. MouseActivate can be overridden in custom control descendants to provide special built-in processing of the OnMouseActivate events. If you leave the default value of MouseActivate as maDefault, the parent control fires the OnMouseActivate event and repeats the process all the way up to the top-level form. If no control in the chain sets the value of MouseActivate, the behavior is the same as if a control set TMouseActivate to maActivate. At any point, a control can set the value of TMouseActivate and the parent processing stops.

Some controls set focus to themselves when they get a button down message so, in some cases, setting maNoActivate appears to have no effect. For instance, a TButton control sets focus to itself in the WM_LBUTTONDOWN message regardless of whether or not TMouseActivate is set to maNoActivate. In this case, setting maNoActivateAndEat will work because the top-level form is not activated and the WM_LBUTTONDOWN message is suppressed.

The HitTest parameter is the hit test value obtained from the WM_NCHITTEST message. See Windows.pas for valid values and their meanings. In most cases this value will be HTCLIENT, which means that the user clicked in the client area of a control. However, in the case of a top-level form, this can take other values such as HTCAPTION, or HTBOTTOM. This allows the code to decide which value to set for MouseActivate, based on the HitTest code. For example, the following code in the OnMouseActivate event handler for a TForm forces the user to click into a non-client area to activate the form:

```
if HitTest = HTCLIENT then
    MouseActivate := maNoActivateAndEat;
```

Even though this event is tied to how Windows processes WM_MOUSEACTIVATE, the actual handling of the message is done at the TControl level, which is the point in the VCL hierarchy where the control has no window handle. The OnMouseActivate event is then simulated since the all TControl descendants also have OnMouseDown, OnMouseMove, and OnMouseUp events. This information is applicable to both VCL Win32 and VCL.NET.

See Also

OnClick

OnMouseMove

OnMouseUp

MouseCapture

29.1.3.3.14 TMaskEdit.OnMouseDown Event

Occurs when the user presses a mouse button with the mouse pointer over a control.

Pascal

```
property OnMouseDown;
```

C++

```
__property OnMouseDown;
```

Description

Use the OnMouseDown event handler to implement any special processing that should occur as a result of pressing a mouse button.

The OnMouseDown event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. X and Y are the pixel coordinates of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
```

```
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  Form1.Canvas.Rectangle(StartX, StartY, X, Y);
  StatusBar1.Panels[0].Text := '';
  StatusBar1.Panels[1].Text := '';
  StatusBar1.Panels[2].Text := '';
  StatusBar1.Panels[3].Text := '';
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if ssLeft in Shift then { make sure button is down }
  begin
    if Y > StartY then
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
      end
    else
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
      end;
    if X > StartX then
      begin
```

```
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
        end
      else
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
        end;
    end;
  end;
```

See Also

OnClick

OnMouseMove

OnMouseUp

MouseCapture

29.1.3.3.15 TMaskEdit.OnMouseEnter Event

Occurs when the user moves the mouse into a control.

Pascal

```
property OnMouseEnter;
```

C++

```
__property OnMouseEnter;
```

Description

Use the OnMouseEnter event handler to implement any special processing that should occur as a result of moving the mouse into a control when the parent top-level form is not active.

29.1.3.3.16 TMaskEdit.OnMouseLeave Event

Occurs when the user moves the mouse outside of a control.

Pascal

```
property OnMouseLeave;
```

C++

```
__property OnMouseLeave;
```

Description

Use the OnMouseLeave event handler to implement any special processing that should occur as a result of moving the mouse outside of a control when the parent top-level form is not active.

29.1.3.3.17 TMaskEdit.OnMouseMove Event

Occurs when the user moves the mouse pointer while the mouse pointer is over a control.

Pascal

```
property OnMouseMove;
```

C++

```
__property OnMouseMove;
```

Description

Use the OnMouseMove event handler to respond when the mouse pointer moves after the control has captured the mouse.

Use the Shift parameter of the OnMouseDown event handler, to determine to the state of the shift keys and mouse buttons. Shift keys are the Shift, Ctrl, and Alt keys or shift key-mouse button combinations. X and Y are pixel coordinates of the new location of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
```

```
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  Form1.Canvas.Rectangle(StartX, StartY, X, Y);
  StatusBar1.Panels[0].Text := '';
  StatusBar1.Panels[1].Text := '';
  StatusBar1.Panels[2].Text := '';
  StatusBar1.Panels[3].Text := '';
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if ssLeft in Shift then { make sure button is down }
  begin
    if Y > StartY then
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
      end
    else
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
      end;
    if X > StartX then
      begin
      StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
      StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
      end
    else
      begin
      StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
      StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
      end;
  end;
end;
```

See Also

OnMouseDown

OnMouseUp

MouseCapture

29.1.3.3.18 TMaskEdit.OnMouseUp Event

Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.

Pascal

```
property OnMouseUp;
```

C++

```
__property OnMouseUp;
```

Description

Use an OnMouseUp event handler to implement special processing when the user releases a mouse button.

The OnMouseUp event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. X and Y are the pixel coordinates of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
```

```
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  Form1.Canvas.Rectangle(StartX, StartY, X, Y);
  StatusBar1.Panels[0].Text := '';
  StatusBar1.Panels[1].Text := '';
  StatusBar1.Panels[2].Text := '';
  StatusBar1.Panels[3].Text := '';
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if ssLeft in Shift then { make sure button is down }
  begin
    if Y > StartY then
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
      end
    else
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
      end;
```

```
      if X > StartX then
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
        end
      else
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
        end;
    end;
  end;
```

See Also

OnMouseDown

OnMouseMove

Responding to a Mouse-up Action

29.1.3.3.19 TMaskEdit.OnStartDock Event

Occurs when the user begins to drag a control with a DragKind of dkDock.

Pascal

```
property OnStartDock;
```

C++

```
__property OnStartDock;
```

Description

Use the OnStartDock event handler to implement special processing when the user starts a drag-and-dock operation by dragging the control.

The OnStartDock event handler can create a TDragDockObjectEx object for the DragObject parameter to specify the appearance of the dragging rectangle and how the dragged control interacts with potential docking sites. If you return TDragDockObjectEx as the drag object, there is no need to call the Free method for the DragObject when dragging is over. If you use TDragDockObject, your application is responsible for freeing the drag object.

If the OnStartDock event handler sets the DragObject parameter to nil (Delphi) or NULL (C++), a TDragDockObject object is automatically created.

Note: This information applies to Win32 applications only.

See Also

OnEndDock

TDragDockObjectEx

DragKind

OnDragOver

OnDockDrop

29.1.3.3.20 TMaskEdit.OnStartDrag Event

Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

Pascal

```
property OnStartDrag;
```

C++

```
__property OnStartDrag;
```

Description

Use the OnStartDrag event handler to implement special processing when the user starts to drag the control or an object it contains. OnStartDrag only occurs if DragKind is dkDrag.

Sender is the control that is about to be dragged, or that contains the object about to be dragged.

The OnStartDrag event handler can create a TDragControlObjectEx instance for the DragObject parameter to specify the drag cursor, or, optionally, a drag image list. If you create a TDragControlObjectEx instance, there is no need to call the Free method for the DragObject when dragging is over. If you create, instead, a TDragControlObject instance, your application is responsible for freeing the drag object instance.

If the OnStartDrag event handler sets the DragObject parameter to nil (Delphi) or NULL (C++), a TDragControlObject object is automatically created and dragging begins on the control itself.

Note: On some controls, such as TRichEdit, the underlying Windows control handles internal drag operations. For these controls, there are no OnStartDrag or OnEndDrag events for drag operations within the control.

See Also

OnEndDrag

TDragControlObjectEx

DragKind

OnDragOver

OnDragDrop

29.2 Types

The following table lists types in this documentation.

Types

Name	Description
TMaskedState (see page 1175)	TMaskedState is a set of flags that describe the current state of a masked edit control.

29.2.1 Mask.TMaskedState Type

TMaskedState is a set of flags that describe the current state of a masked edit control.

Pascal

```
TMaskedState = set of (msMasked, msReEnter, msDBSetText);
```

C++

```
(msMasked, msReEnter, msDBSetText) TMaskedState;
```

File

Mask

Description

The following table indicates the meaning of the various TMaskedState flags:

Value	Meaning
msMasked	A mask has been assigned. MaskState includes msMasked when IsMasked is true.
msReEnter	An invalid character has been rejected and the next edit should be a valid replacement character.
msDBSetText	The text comes from a database field and should not be validated.

30 Masks Namespace

This is namespace Masks.

Topics

Name	Description
Classes (see page 1178)	The following table lists classes in this documentation.
Functions (see page 1182)	The following table lists functions in this documentation.

Classes

	Name	Description
	EMaskException (see page 1178)	EMaskException is the exception class for errors that occur when comparing a file name to a mask.
	TMask (see page 1178)	TMask allows the comparison of strings containing wildcards to a file mask.

Functions

	Name	Description
	MatchesMask (see page 1182)	Indicates whether a file name conforms to the format specified by a filter string.

30.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	EMaskException (see page 1178)	EMaskException is the exception class for errors that occur when comparing a file name to a mask.
	TMask (see page 1178)	TMask allows the comparison of strings containing wildcards to a file mask.

30.1.1 EMaskException Class

EMaskException is the exception class for errors that occur when comparing a file name to a mask.

Class Hierarchy

Pascal

```
EMaskException = class(Exception);
```

C++

```
class EMaskException : public Exception;
```

File

Masks

Description

EMaskException is raised when the TMask (see page 1178) object or the global MatchesMask (see page 1182) function encounters an improperly formed mask string.

See Also

TMask (see page 1178)

MatchesMask (see page 1182)

30.1.2 TMask Class

TMask allows the comparison of strings containing wildcards to a file mask.

Class Hierarchy

Pascal

```
TMask = class;
```

C++

```
class TMask;
```

File

Masks

Description

Use TMask to determine if a string matches the format described by a mask. The mask contains literal characters, wildcards that can match any character, and sets that match a single character that is included in the set.

Note: Do not confuse TMask with the EditMask of a field or masked edit object. While both are used for comparing strings to a symbolic description of valid values, the special mask symbols and matching rules are completely different.

See Also

TCustomMaskEdit (see page 1080)

TField

30.1.2.1 TMask Members

The following tables list the members exposed by TMask.

Public Methods

	Name	Description
	Create (see page 1179)	Creates an instance of TMask (see page 1178) for a given mask.
	Destroy (see page 1180)	Destroys an instance of TMask (see page 1178).
	Matches (see page 1180)	Indicates whether a specified string matches the mask.

30.1.2.2 TMask.Create Constructor

Creates an instance of TMask (see page 1178) for a given mask.

Pascal

```
constructor Create(const MaskValue: string);
```

C++

```
__fastcall TMask(const UnicodeString MaskValue);
```

Description

Call Create to create a mask object for comparing strings to the mask given by the MaskValue parameter. A valid mask consists of literal characters, sets, and wildcards.

Each literal character must match a single character in the string. The comparison to literal characters is case-insensitive.

Each set begins with an opening bracket ([) and ends with a closing bracket (]). Between the brackets are the elements of the set. Each element is a literal character or a range. Ranges are specified by an initial value, a dash (-), and a final value. Do not use spaces or commas to separate the elements of the set. A set must match a single character in the string. The character matches the set if it is the same as one of the literal characters in the set, or if it is in one of the ranges in the set. A character is in a range if it matches the initial value, the final value, or falls between the two values. All comparisons are case-insensitive. If the first character after the opening bracket of a set is an exclamation point (!), then the set matches any character that is not in the set.

Wildcards are asterisks (*) or question marks (?). An asterisk matches any number of characters. A question mark matches a

single arbitrary character.

See Also

Matches (see page 1180)

30.1.2.3 TMask.Destroy Destructor

Destroys an instance of TMask (see page 1178).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TMask();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free checks that the TMask (see page 1178) reference is not nil, and only then calls Destroy.

Destroy frees the memory used to store a parsed image of the mask.

See Also

System::TObject::Free

30.1.2.4 TMask Methods

The methods of the TMask class are listed here.

Public Methods

	Name	Description
	Matches (see page 1180)	Indicates whether a specified string matches the mask.

30.1.2.4.1 TMask.Matches Method

Indicates whether a specified string matches the mask.

Pascal

```
function Matches(const Filename: string): Boolean;
```

C++

```
__fastcall Boolean Matches(const UnicodeString Filename);
```

Description

Call Matches to compare the Filename parameter to the mask that describes valid values. Matches returns true if the string matches the mask. Matches returns false if the string does not match the mask. Matches raises an exception if the mask is syntactically invalid.

The mask is set when the TMask (see page 1178) object is created.

Note: Filename parameter does not need to be a file name. TMask (see page 1178) can be used to check strings against any syntactically correct mask.

See Also

Create (see page 1179)

30.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	MatchesMask (see page 1182)	Indicates whether a file name conforms to the format specified by a filter string.

30.2.1 Masks.MatchesMask Function

Indicates whether a file name conforms to the format specified by a filter string.

Pascal

```
function MatchesMask(const Filename: string; const Mask: string): Boolean;
```

C++

```
Boolean MatchesMask(const UnicodeString Filename, const UnicodeString Mask);
```

File

Masks

Description

Call MatchesMask to check the Filename parameter using the Mask (see page 1079) parameter to describe valid values. A valid mask consists of literal characters, sets, and wildcards.

Each literal character must match a single character in the string. The comparison to literal characters is case-insensitive.

Each set begins with an opening bracket ([) and ends with a closing bracket (]). Between the brackets are the elements of the set. Each element is a literal character or a range. Ranges are specified by an initial value, a dash (-), and a final value. Do not use spaces or commas to separate the elements of the set. A set must match a single character in the string. The character matches the set if it is the same as one of the literal characters in the set, or if it is in one of the ranges in the set. A character is in a range if it matches the initial value, the final value, or falls between the two values. All comparisons are case-insensitive. If the first character after the opening bracket of a set is an exclamation point (!), then the set matches any character that is not in the set.

Wildcards are asterisks (*) or question marks (?). An asterisk matches any number of characters. A question mark matches a single arbitrary character.

MatchesMask returns true if the string matches the mask. MatchesMask returns false if the string does not match the mask. MatchesMask raises an exception if the mask is syntactically invalid.

Note: The Filename parameter does not need to be a file name. MatchesMask can be used to check strings against any syntactically correct mask.

See Also

TMask (see page 1178)

31 MaskUtils Namespace

This is namespace MaskUtils.

Topics

Name	Description
Functions (see page 1185)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 1192)	The following table lists structs, records, enums in this documentation.
Types (see page 1193)	The following table lists types in this documentation.
Variables (see page 1196)	The following table lists variables in this documentation.
Constants (see page 1198)	The following table lists constants in this documentation.

Constants

Name	Description
mDirLiteral (see page 1198)	mDirLiteral is used internally.
mDirLowerCase (see page 1198)	mDirLowerCase is used internally.
mDirReverse (see page 1199)	mDirReverse is used internally.
mDirUpperCase (see page 1199)	mDirUpperCase is used internally.
mMskAlpha (see page 1200)	mMskAlpha is used internally.
mMskAlphaNum (see page 1200)	mMskAlphaNum is used internally.
mMskAlphaNumOpt (see page 1200)	mMskAlphaNumOpt is used internally.
mMskAlphaOpt (see page 1201)	mMskAlphaOpt is used internally.
mMskAscii (see page 1201)	mMskAscii is used internally.
mMskAsciiOpt (see page 1202)	mMskAsciiOpt is used internally.
mMskDateSeparator (see page 1202)	mMskDateSeparator is used internally.
mMskNumeric (see page 1202)	mMskNumeric is used internally.
mMskNumericOpt (see page 1203)	mMskNumericOpt is used internally.
mMskNumSymOpt (see page 1203)	mMskNumSymOpt is used internally.
mMskTimeSeparator (see page 1204)	mMskTimeSeparator is used internally.

Functions

	Name	Description
	FormatMaskText (see page 1185)	Returns a string formatted using an edit mask.
	IsLiteralChar (see page 1187)	IsLiteralChar is used internally.
	MaskDoFormatText (see page 1187)	MaskDoFormatText is used internally.
	MaskGetCharType (see page 1187)	MaskGetCharType is used internally.
	MaskGetCurrentDirectives (see page 1188)	MaskGetCurrentDirectives is used internally.

	MaskGetFldSeparator (see page 1188)	MaskGetFldSeparator is used internally.
	MaskGetMaskBlank (see page 1189)	MaskGetMaskBlank is used internally.
	MaskGetMaskSave (see page 1189)	MaskGetMaskSave is used internally.
	MaskIntlLiteralToChar (see page 1189)	MaskIntlLiteralToChar is used internally.
	MaskOffsetToOffset (see page 1190)	MaskOffsetToOffset is used internally.
	MaskOffsetToWideOffset (see page 1190)	MaskOffsetToWideOffset is used internally.
	OffsetToMaskOffset (see page 1191)	OffsetToMaskOffset is used internally.
	PadInputLiterals (see page 1191)	PadInputLiterals is used internally.

Structs, Records, Enums

	Name	Description
	TMaskCharType (see page 1192)	TMaskCharType is used internally.

Types

Name	Description
TEditMask (see page 1193)	Represents the mask that validates and formats user input.
TMaskDirectives (see page 1194)	TMaskDirectives is used internally.
TMaskedText (see page 1195)	TMaskedText is the type for the text that is typed into a masked edit control.

Variables

Name	Description
DefaultBlank (see page 1196)	DefaultBlank is used internally.
MaskFieldSeparator (see page 1196)	MaskFieldSeparator is used internally.
MaskNoSave (see page 1197)	MaskNoSave is used internally.

31.1 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	FormatMaskText (see page 1185)	Returns a string formatted using an edit mask.
	IsLiteralChar (see page 1187)	IsLiteralChar is used internally.
	MaskDoFormatText (see page 1187)	MaskDoFormatText is used internally.
	MaskGetCharType (see page 1187)	MaskGetCharType is used internally.
	MaskGetCurrentDirectives (see page 1188)	MaskGetCurrentDirectives is used internally.
	MaskGetFldSeparator (see page 1188)	MaskGetFldSeparator is used internally.
	MaskGetMaskBlank (see page 1189)	MaskGetMaskBlank is used internally.
	MaskGetMaskSave (see page 1189)	MaskGetMaskSave is used internally.
	MaskIntlLiteralToChar (see page 1189)	MaskIntlLiteralToChar is used internally.
	MaskOffsetToOffset (see page 1190)	MaskOffsetToOffset is used internally.
	MaskOffsetToWideOffset (see page 1190)	MaskOffsetToWideOffset is used internally.
	OffsetToMaskOffset (see page 1191)	OffsetToMaskOffset is used internally.
	PadInputLiterals (see page 1191)	PadInputLiterals is used internally.

31.1.1 MaskUtils.FormatMaskText Function

Returns a string formatted using an edit mask.

Pascal

```
function FormatMaskText(const EditMask: string; const Value: string): string;
```

C++

```
UnicodeString FormatMaskText(const UnicodeString EditMask, const UnicodeString Value);
```

File

MaskUtils

Description

Call FormatMaskText to apply the mask specified by the EditMask parameter to the text string specified by the Value parameter. The edit mask string consists of three fields with semicolons separating them. The first part of the mask is the mask itself. The second part is the character that determines whether the literal characters of the mask are matched to characters in the Value parameter or are inserted into the Value string. The third part of the mask is the character used to represent missing characters in the mask.

These are the special characters used in the first field of the mask:

Character	Meaning in mask
!	If a ! character appears in the mask, optional characters are represented in the returned string as leading blanks. If a ! character is not present, optional characters are represented in the returned string as trailing blanks.
>	If a > character appears in the mask, all characters that follow are in uppercase until the end of the mask or until a < character is encountered.

<	If a < character appears in the mask, all characters that follow are in lowercase until the end of the mask or until a > character is encountered.
<>	If these two characters appear together in a mask, no case checking is done and the data is formatted with the case present in the Value parameter.
\	The character that follows a \ character is a literal character. Use this character to use any of the mask special characters as a literal.
L	The L character requires an alphabetic character only in this position. For the US, this is A-Z, a-z.
l	The l character permits only an alphabetic character in this position, but doesn't require it.
A	The A character requires an alphanumeric character only in this position. For the US, this is A-Z, a-z, 0-9.
a	The a character permits an alphanumeric character in this position, but doesn't require it.
C	The C character requires an arbitrary character in this position.
c	The c character permits an arbitrary character in this position, but doesn't require it.
0	The 0 character requires a numeric character only in this position.
9	The 9 character permits a numeric character in this position, but doesn't require it.
#	The # character permits a numeric character or a plus or minus sign in this position, but doesn't require it.
:	The : character is used to separate hours, minutes, and seconds in times. If the character that separates hours, minutes, and seconds is different in the regional settings of the Control Panel, that character is substituted in the returned string.
/	The / character is used to separate months, days, and years in dates. If the character that separates months, days, and years is different in the regional settings of the Control Panel, that character is substituted in the returned string.
;	The ; character is used to separate the three fields of the mask.
_	The _ character automatically inserts spaces into the returned string.

Any character that does not appear in the preceding table can appear in the first part of the mask as a literal character. Literal characters are inserted automatically if the second field of the mask is 0, or matched to characters in the Value parameter if the second field is any other value. The special mask characters can also appear as literal characters if preceded by a backslash character (\).

The second field of the mask is a single character that indicates whether literal characters from the mask are included in the Value parameter. For example, the mask for a telephone number with area code could be the following string:

(000)_000-0000;0;*

The 0 in the second field indicates that the Value parameter should consist of the 10 digits of the phone number, rather than the 14 characters that make up the final formatted string.

A 0 in the second field indicates that literals are inserted into the Value string, any other character indicates that they should be included. The character to indicate whether literals should be included can be changed by changing the MaskNoSave (see page 1197) constant that is declared in the MaskUtils (see page 1183) unit.

The third field of the mask is the character that appears in the returned string for blanks (characters that do not appear in Value). By default, this is the same as the character that stands for literal spaces. The two characters appear the same in the returned string.

Note: When working with multibyte character sets, each special mask character represents a single byte. To specify double-byte characters using the L, l, A, a, C, or c specifiers, the mask characters must be doubled as well. For example, LL would represent two single-byte alphabetic characters or a one double-byte character. Only single-byte literal characters are supported.

See Also

TCustomMaskEdit (see page 1080)

TField

31.1.2 MaskUtils.IsLiteralChar Function

IsLiteralChar is used internally.

Pascal

```
function IsLiteralChar(const EditMask: string; Offset: Integer): Boolean;
```

C++

```
Boolean IsLiteralChar(const UnicodeString EditMask, int Offset);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call IsLiteralChar directly.

See Also

FormatMaskText (see page 1185)

31.1.3 MaskUtils.MaskDoFormatText Function

MaskDoFormatText is used internally.

Pascal

```
function MaskDoFormatText(const EditMask: string; const Value: string; Blank: Char): string;
```

C++

```
UnicodeString MaskDoFormatText(const UnicodeString EditMask, const UnicodeString Value, Char
Blank);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call MaskDoFormatText directly.

See Also

FormatMaskText (see page 1185)

31.1.4 MaskUtils.MaskGetCharType Function

MaskGetCharType is used internally.

Pascal

```
function MaskGetCharType(const EditMask: string; MaskOffset: Integer): TMaskCharType;
```

C++

```
TMaskCharType MaskGetCharType(const UnicodeString EditMask, int MaskOffset);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call MaskGetCharType directly.

See Also

FormatMaskText (see page 1185)

31.1.5 MaskUtils.MaskGetCurrentDirectives Function

MaskGetCurrentDirectives is used internally.

Pascal

```
function MaskGetCurrentDirectives(const EditMask: string; MaskOffset: Integer):
TMaskDirectives;
```

C++

```
TMaskDirectives MaskGetCurrentDirectives(const UnicodeString EditMask, int MaskOffset);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call MaskGetCurrentDirectives directly.

See Also

FormatMaskText (see page 1185)

31.1.6 MaskUtils.MaskGetFldSeparator Function

MaskGetFldSeparator is used internally.

Pascal

```
function MaskGetFldSeparator(const EditMask: string): Integer;
```

C++

```
int MaskGetFldSeparator(const UnicodeString EditMask);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call MaskGetFldSeparator directly.

See Also

FormatMaskText (see page 1185)

31.1.7 MaskUtils.MaskGetMaskBlank Function

MaskGetMaskBlank is used internally.

Pascal

```
function MaskGetMaskBlank(const EditMask: string): Char;
```

C++

```
Char MaskGetMaskBlank(const UnicodeString EditMask);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call MaskGetMaskBlank directly.

See Also

FormatMaskText (see page 1185)

31.1.8 MaskUtils.MaskGetMaskSave Function

MaskGetMaskSave is used internally.

Pascal

```
function MaskGetMaskSave(const EditMask: string): Boolean;
```

C++

```
Boolean MaskGetMaskSave(const UnicodeString EditMask);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call MaskGetMaskSave directly.

See Also

FormatMaskText (see page 1185)

31.1.9 MaskUtils.MaskIntlLiteralToChar Function

MaskIntlLiteralToChar is used internally.

Pascal

```
function MaskIntlLiteralToChar(IChar: Char): Char;
```

C++

```
Char MaskIntlLiteralToChar(Char IChar);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call MaskIntlLiteralToChar directly.

See Also

FormatMaskText (see page 1185)

31.1.10 MaskUtils.MaskOffsetToOffset Function

MaskOffsetToOffset is used internally.

Pascal

```
function MaskOffsetToOffset(const EditMask: String; MaskOffset: Integer): Integer;
```

C++

```
int MaskOffsetToOffset(const UnicodeString EditMask, int MaskOffset);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call MaskOffsetToOffset directly.

See Also

FormatMaskText (see page 1185)

31.1.11 MaskUtils.MaskOffsetToWideOffset Function

MaskOffsetToWideOffset is used internally.

Pascal

```
function MaskOffsetToWideOffset(const EditMask: String; MaskOffset: Integer): Integer;
```

C++

```
int MaskOffsetToWideOffset(const UnicodeString EditMask, int MaskOffset);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call MaskOffsetToWideOffset directly.

See Also

FormatMaskText (see page 1185)

31.1.12 MaskUtils.OffsetToMaskOffset Function

OffsetToMaskOffset is used internally.

Pascal

```
function OffsetToMaskOffset(const EditMask: string; Offset: Integer): Integer;
```

C++

```
int OffsetToMaskOffset(const UnicodeString EditMask, int Offset);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call OffsetToMaskOffset directly.

See Also

FormatMaskText (see page 1185)

31.1.13 MaskUtils.PadInputLiterals Function

PadInputLiterals is used internally.

Pascal

```
function PadInputLiterals(const EditMask: String; const Value: string; Blank: Char): string;
```

C++

```
UnicodeString PadInputLiterals(const UnicodeString EditMask, const UnicodeString Value, Char
Blank);
```

File

MaskUtils

Description

This function is used internally by FormatMaskText (see page 1185). Do not call PadInputLiterals directly.

See Also

FormatMaskText (see page 1185)

31.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TMaskCharType (see page 1192)	TMaskCharType is used internally.

31.2.1 MaskUtils.TMaskCharType Enumeration

TMaskCharType is used internally.

Pascal

```
TMaskCharType = (
  mcNone,
  mcLiteral,
  mcIntlLiteral,
  mcDirective,
  mcMask,
  mcMaskOpt,
  mcFieldSeparator,
  mcField
);
```

C++

```
enum TMaskCharType {
  mcNone,
  mcLiteral,
  mcIntlLiteral,
  mcDirective,
  mcMask,
  mcMaskOpt,
  mcFieldSeparator,
  mcField
};
```

File

MaskUtils

Description

This record is used internally by FormatMaskText (see page 1185). Do not use TMaskCharType directly.

See Also

FormatMaskText (see page 1185)

31.3 Types

The following table lists types in this documentation.

Types

Name	Description
TEditMask (see page 1193)	Represents the mask that validates and formats user input.
TMaskDirectives (see page 1194)	TMaskDirectives is used internally.
TMaskedText (see page 1195)	TMaskedText is the type for the text that is typed into a masked edit control.

31.3.1 MaskUtils.TEditMask Type

Represents the mask that validates and formats user input.

Pascal

```
TEditMask = type string;
```

C++

```
type string TEditMask;
```

File

MaskUtils

Description

TEditMask is a string that consists of three fields with semicolons separating them. The first part of the mask is the mask itself. The second part is the character that determines whether the literal characters of a mask are saved as part of the data. The third part of the mask is the character used to represent unentered characters in the mask.

These are the special characters used in the first field of the mask:

Character	Meaning in mask
!	If a ! character appears in the mask, optional characters are represented in the text as leading blanks. If a ! character is not present, optional characters are represented in the text as trailing blanks.
>	If a > character appears in the mask, all characters that follow are in uppercase until the end of the mask or until a < character is encountered.
<	If a < character appears in the mask, all characters that follow are in lowercase until the end of the mask or until a > character is encountered.
<>	If these two characters appear together in a mask, no case checking is done and the data is formatted with the case the user uses to enter the data.
\	The character that follows a \ character is a literal character. Use this character to use any of the mask special characters as a literal in the data.
L	The L character requires an alphabetic character only in this position. For the US, this is A-Z, a-z.
l	The l character permits only an alphabetic character in this position, but doesn't require it.
A	The A character requires an alphanumeric character only in this position. For the US, this is A-Z, a-z, 0-9.
a	The a character permits an alphanumeric character in this position, but doesn't require it.

C	The C character requires an arbitrary character in this position.
c	The c character permits an arbitrary character in this position, but doesn't require it.
0	The 0 character requires a numeric character only in this position.
9	The 9 character permits a numeric character in this position, but doesn't require it.
#	The # character permits a numeric character or a plus or minus sign in this position, but doesn't require it.
:	The : character is used to separate hours, minutes, and seconds in times. If the character that separates hours, minutes, and seconds is different in the regional settings of the Control Panel utility on your computer system, that character is used instead.
/	The / character is used to separate months, days, and years in dates. If the character that separates months, days, and years is different in the regional settings of the Control Panel utility on your computer system, that character is used instead.
;	The ; character is used to separate the three fields of the mask.
_	The _ character automatically inserts spaces into the text. When the user enters characters in the field, the cursor skips the _ character.

Any character that does not appear in the preceding table can appear in the first part of the mask as a literal character. Literal characters must be matched exactly in the edit control. They are inserted automatically, and the cursor skips over them during editing. The special mask characters can also appear as literal characters if preceded by a backslash character (\).

The second field of the mask is a single character that indicates whether literal characters from the mask should be included as part of the text for the edit control. For example, the mask for a telephone number with area code could be the following string:

(000)_000-0000;0;*

The 0 in the second field indicates that the Text property for the edit control would consist of the 10 digits that were entered, rather than the 14 characters that make up the telephone number as it appears in the edit control.

A 0 in the second field indicates that literals should not be included, any other character indicates that they should be included. The character that indicates whether literals should be included can be changed in the Edit Mask (see page 1079) property editor, or programmatically by changing the MaskNoSave (see page 1197) typed constant.

The third field of the mask is the character that appears in the edit control for blanks (characters that have not been entered). By default, this is the same as the character that stands for literal spaces. The two characters appear the same in an edit window. However, when a user edits the text in a masked edit control, the cursor selects each blank character in turn, and skips over the space character.

Note: When working with multibyte character sets, each special mask character represents a single byte. To specify multi-byte characters using the L, l, A, a, C, or c specifiers, the mask characters must be duplicated as well. For example, LL would represent two single-byte alphabetic characters or a one double-byte character. Only single-byte literal characters are supported.

31.3.2 MaskUtils.TMaskDirectives Type

TMaskDirectives is used internally.

Pascal

```
TMaskDirectives = set of (mdReverseDir, mdUpperCase, mdLowerCase, mdLiteralChar);
```

C++

```
(mdReverseDir, mdUpperCase, mdLowerCase, mdLiteralChar) TMaskDirectives;
```

File

MaskUtils

Description

This record is used internally by FormatMaskText (see page 1185). Do not use TMaskDirectives directly.

See Also

FormatMaskText (see page 1185)

31.3.3 MaskUtils.TMaskedText Type

TMaskedText is the type for the text that is typed into a masked edit control.

Pascal

```
TMaskedText = type string;
```

C++

```
type string TMaskedText;
```

File

MaskUtils

Description

TMaskedText is the value of a masked edit control's text before the mask is applied.

31.4 Variables

The following table lists variables in this documentation.

Variables

Name	Description
DefaultBlank (see page 1196)	DefaultBlank is used internally.
MaskFieldSeparator (see page 1196)	MaskFieldSeparator is used internally.
MaskNoSave (see page 1197)	MaskNoSave is used internally.

31.4.1 MaskUtils.DefaultBlank Variable

DefaultBlank is used internally.

Pascal

```
DefaultBlank: Char = '_';
```

C++

```
Char DefaultBlank = '_';
```

File

MaskUtils

Description

This variable is used internally by FormatMaskText (see page 1185). Do not use DefaultBlank directly.

See Also

FormatMaskText (see page 1185)

31.4.2 MaskUtils.MaskFieldSeparator Variable

MaskFieldSeparator is used internally.

Pascal

```
MaskFieldSeparator: Char = ';';
```

C++

```
Char MaskFieldSeparator = ';';
```

File

MaskUtils

Description

This variable is used internally by FormatMaskText (see page 1185). Do not use MaskFieldSeparator directly.

See Also

FormatMaskText (see page 1185)

31.4.3 MaskUtils.MaskNoSave Variable

MaskNoSave is used internally.

Pascal

```
MaskNoSave: Char = '0';
```

C++

```
Char MaskNoSave = '0';
```

File

MaskUtils

Description

This variable is used internally by FormatMaskText (see page 1185). Do not use MaskNoSave directly.

See Also

FormatMaskText (see page 1185)

31.5 Constants

The following table lists constants in this documentation.

Constants

Name	Description
mDirLiteral (see page 1198)	mDirLiteral is used internally.
mDirLowerCase (see page 1198)	mDirLowerCase is used internally.
mDirReverse (see page 1199)	mDirReverse is used internally.
mDirUpperCase (see page 1199)	mDirUpperCase is used internally.
mMskAlpha (see page 1200)	mMskAlpha is used internally.
mMskAlphaNum (see page 1200)	mMskAlphaNum is used internally.
mMskAlphaNumOpt (see page 1200)	mMskAlphaNumOpt is used internally.
mMskAlphaOpt (see page 1201)	mMskAlphaOpt is used internally.
mMskAscii (see page 1201)	mMskAscii is used internally.
mMskAsciiOpt (see page 1202)	mMskAsciiOpt is used internally.
mMskDateSeparator (see page 1202)	mMskDateSeparator is used internally.
mMskNumeric (see page 1202)	mMskNumeric is used internally.
mMskNumericOpt (see page 1203)	mMskNumericOpt is used internally.
mMskNumSymOpt (see page 1203)	mMskNumSymOpt is used internally.
mMskTimeSeparator (see page 1204)	mMskTimeSeparator is used internally.

31.5.1 MaskUtils.mDirLiteral Constant

mDirLiteral is used internally.

Pascal

```
mDirLiteral = '\';
```

C++

```
mDirLiteral = '\';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mDirLiteral directly.

See Also

FormatMaskText (see page 1185)

31.5.2 MaskUtils.mDirLowerCase Constant

mDirLowerCase is used internally.

Pascal

```
mDirLowerCase = '<';
```

C++

```
mDirLowerCase = '<';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mDirLowerCase directly.

See Also

FormatMaskText (see page 1185)

31.5.3 MaskUtils.mDirReverse Constant

mDirReverse is used internally.

Pascal

```
mDirReverse = '!';
```

C++

```
mDirReverse = '!';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mDirReverse directly.

See Also

FormatMaskText (see page 1185)

31.5.4 MaskUtils.mDirUpperCase Constant

mDirUpperCase is used internally.

Pascal

```
mDirUpperCase = '>';
```

C++

```
mDirUpperCase = '>';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mDirUpperCase directly.

See Also

FormatMaskText (see page 1185)

31.5.5 MaskUtils.mMskAlpha Constant

mMskAlpha is used internally.

Pascal

```
mMskAlpha = 'L';
```

C++

```
mMskAlpha = 'L';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mMskAlpha directly.

See Also

FormatMaskText (see page 1185)

31.5.6 MaskUtils.mMskAlphaNum Constant

mMskAlphaNum is used internally.

Pascal

```
mMskAlphaNum = 'A';
```

C++

```
mMskAlphaNum = 'A';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mMskAlphaNum directly.

See Also

FormatMaskText (see page 1185)

31.5.7 MaskUtils.mMskAlphaNumOpt Constant

mMskAlphaNumOpt is used internally.

Pascal

```
mMskAlphaNumOpt = 'a';
```

C++

```
mMskAlphaNumOpt = 'a';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mMskAlphaNumOpt directly.

See Also

FormatMaskText (see page 1185)

31.5.8 MaskUtils.mMskAlphaOpt Constant

mMskAlphaOpt is used internally.

Pascal

```
mMskAlphaOpt = 'l';
```

C++

```
mMskAlphaOpt = 'l';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mMskAlphaOpt directly.

See Also

FormatMaskText (see page 1185)

31.5.9 MaskUtils.mMskAscii Constant

mMskAscii is used internally.

Pascal

```
mMskAscii = 'C';
```

C++

```
mMskAscii = 'C';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mMskAscii directly.

See Also

FormatMaskText (see page 1185)

31.5.10 MaskUtils.mMskAsciiOpt Constant

mMskAsciiOpt is used internally.

Pascal

```
mMskAsciiOpt = 'c';
```

C++

```
mMskAsciiOpt = 'c';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mMskAsciiOpt directly.

See Also

FormatMaskText (see page 1185)

31.5.11 MaskUtils.mMskDateSeparator Constant

mMskDateSeparator is used internally.

Pascal

```
mMskDateSeparator = '/';
```

C++

```
mMskDateSeparator = '/';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mMskDateSeparator directly.

See Also

FormatMaskText (see page 1185)

31.5.12 MaskUtils.mMskNumeric Constant

mMskNumeric is used internally.

Pascal

```
mMskNumeric = '0';
```

C++

```
mMskNumeric = '0';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mMskNumeric directly.

See Also

FormatMaskText (see page 1185)

31.5.13 MaskUtils.mMskNumericOpt Constant

mMskNumericOpt is used internally.

Pascal

```
mMskNumericOpt = '9';
```

C++

```
mMskNumericOpt = '9';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mMskNumericOpt directly.

See Also

FormatMaskText (see page 1185)

31.5.14 MaskUtils.mMskNumSymOpt Constant

mMskNumSymOpt is used internally.

Pascal

```
mMskNumSymOpt = '#';
```

C++

```
mMskNumSymOpt = '#';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mMskNumSymOpt directly.

See Also

FormatMaskText (see page 1185)

31.5.15 MaskUtils.mMskTimeSeparator Constant

mMskTimeSeparator is used internally.

Pascal

```
mMskTimeSeparator = ':';
```

C++

```
mMskTimeSeparator = ':';
```

File

MaskUtils

Description

This constant is used internally by FormatMaskText (see page 1185). Do not use mMskTimeSeparator directly.

See Also

FormatMaskText (see page 1185)

32 Math Namespace

This is namespace Math.

Topics

Name	Description
Classes (see page 1209)	The following table lists classes in this documentation.
Functions (see page 1211)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 1263)	The following table lists structs, records, enums in this documentation.
Types (see page 1267)	The following table lists types in this documentation.
Constants (see page 1270)	The following table lists constants in this documentation.

Classes

	Name	Description
	EInvalidArgument (see page 1209)	EInvalidArgument is the exception class for out-of-range parameters in specialized math and accounting functions.

Constants

Name	Description
Infinity (see page 1270)	Infinity represents positive infinity.
MaxComp (see page 1271)	MaxComp defines the maximum absolute value for a Comp variable.
MaxDouble (see page 1271)	MaxDouble defines the maximum absolute value for a Double variable.
MaxExtended (see page 1271)	MaxExtended defines the maximum absolute value for an Extended variable.
MaxSingle (see page 1272)	MaxSingle defines the maximum absolute value for a Single variable.
MinComp (see page 1272)	MinComp defines the smallest absolute value for a Comp variable.
MinDouble (see page 1273)	MinDouble defines the smallest absolute value for a Double variable.
MinExtended (see page 1273)	MinExtended defines the smallest absolute value for an Extended variable.
MinSingle (see page 1273)	MinSingle defines the smallest absolute value for a Single variable.
NaN (see page 1274)	NaN represents a value that cannot be represented as a valid number.
NegativeValue (see page 1274)	NegativeValue defines a negative number result from using Sign (see page 1255).
NegInfinity (see page 1275)	NegInfinity represents negative infinity.
PositiveValue (see page 1275)	PositiveValue defines a positive number result from using Sign (see page 1255).
ZeroValue (see page 1276)	ZeroValue defines a zero value number result from using Sign (see page 1255).

Functions

	Name	Description
	ArcCos (see page 1213)	Calculates the inverse cosine of a given number.
	ArcCos (see page 1213)	Calculates the inverse cosine of a given number.
	ArcCos (see page 1213)	Calculates the inverse cosine of a given number.
	ArcCosh (see page 1214)	Calculates the inverse hyperbolic cosine of a given number.
	ArcCot (see page 1214)	Calculates the inverse cotangent of a given number.
	ArcCotH (see page 1215)	Calculates the inverse hyperbolic cotangent of a given number.
	ArcCsc (see page 1215)	Calculates the inverse cosecant of a given number.
	ArcCscH (see page 1216)	Calculates the inverse hyperbolic cosecant of a given number.
	ArcSec (see page 1216)	Calculates the inverse secant of a given number.
	ArcSecH (see page 1216)	Calculates the inverse hyperbolic secant of a given number.
	ArcSin (see page 1217)	Calculates the inverse sine of a given number.
	ArcSin (see page 1217)	Calculates the inverse sine of a given number.
	ArcSin (see page 1217)	Calculates the inverse sine of a given number.
	ArcSinh (see page 1217)	Calculates the inverse hyperbolic sine of a given number.
	ArcTan2 (see page 1218)	Calculates the arctangent angle and quadrant of a given number.
	ArcTanh (see page 1218)	Calculates the inverse hyperbolic tangent of a given number.
	Ceil (see page 1219)	Rounds variables up toward positive infinity.
	ClearExceptions (see page 1219)	Clears exception bits in FPU status word.
	CompareValue (see page 1220)	Returns the relationship between two numeric values.
	CompareValue (see page 1220)	Returns the relationship between two numeric values.
	CompareValue (see page 1220)	Returns the relationship between two numeric values.
	CompareValue (see page 1220)	Returns the relationship between two numeric values.
	CompareValue (see page 1220)	Returns the relationship between two numeric values.
	Cosecant (see page 1220)	Returns the cosecant of an angle.
	Cosh (see page 1221)	Calculates the hyperbolic cosine of an angle.
	Cot (see page 1221)	Calculates the cotangent of an angle.
	Cotan (see page 1222)	Calculates the cotangent of an angle.
	CotH (see page 1222)	Calculates the hyperbolic cotangent of an angle.
	Csc (see page 1223)	Returns the cosecant of an angle.
	CscH (see page 1223)	Returns the hyperbolic cosecant of an angle.
	CycleToDeg (see page 1224)	Converts an angle measurement from cycles to degrees.
	CycleToGrad (see page 1224)	Converts an angle measurement from cycles to grads.
	CycleToRad (see page 1224)	Converts an angle measurement from cycles to radians.
	DegToCycle (see page 1225)	Returns the value of a degree measurement expressed in cycles.
	DegToGrad (see page 1225)	Returns the value of a degree measurement expressed in grads.
	DegToRad (see page 1226)	Returns the value of a degree measurement expressed in radians.
	DivMod (see page 1226)	Returns the result of an integer division, including the remainder.
	DoubleDecliningBalance (see page 1226)	Calculates the depreciation of an asset using the double-declining balance method.
	EnsureRange (see page 1227)	Returns the closest value to a specified value within a specified range.
	EnsureRange (see page 1227)	Returns the closest value to a specified value within a specified range.
	EnsureRange (see page 1227)	Returns the closest value to a specified value within a specified range.
	Floor (see page 1227)	Rounds variables toward negative infinity.
	Frexp (see page 1228)	Separates the Mantissa and Exponent of X
	FutureValue (see page 1228)	Calculates the future value of an investment.
	GetExceptionMask (see page 1229)	Returns the exception mask from the FPU control word.
	GetPrecisionMode (see page 1229)	Returns the FPU precision control mode.
	GetRoundMode (see page 1230)	Returns the FPU rounding mode.
	GradToCycle (see page 1231)	Converts grad measurements to cycles.
	GradToDeg (see page 1231)	Converts grad measurements to degrees.
	GradToRad (see page 1231)	Converts grad measurements to radians.
	Hypot (see page 1232)	Calculates the length of the hypotenuse.

	IfThen (see page 1232)	Conditionally returns one of two specified values.
	IfThen (see page 1232)	Conditionally returns one of two specified values.
	IfThen (see page 1232)	Conditionally returns one of two specified values.
	InRange (see page 1232)	Indicates whether a value falls within a specified range.
	InRange (see page 1232)	Indicates whether a value falls within a specified range.
	InRange (see page 1232)	Indicates whether a value falls within a specified range.
	InterestPayment (see page 1233)	Calculates the interest portion of a loan payment.
	InterestRate (see page 1234)	Returns the interest rate required to increase PresentValue (see page 1247) to FutureValue (see page 1228).
	InternalRateOfReturn (see page 1234)	Calculates the internal rate of return for an investment.
	IntPower (see page 1235)	Calculates the integral power of a base value.
	IsInfinite (see page 1235)	Indicates when a variable or expression represents an infinite value.
	IsNan (see page 1236)	Indicates when a variable or expression does not evaluate to a numeric value.
	IsNan (see page 1236)	Indicates when a variable or expression does not evaluate to a numeric value.
	IsNan (see page 1236)	Indicates when a variable or expression does not evaluate to a numeric value.
	IsZero (see page 1236)	Indicates when a floating-point variable or expression evaluates to zero, or very close to zero.
	IsZero (see page 1236)	Indicates when a floating-point variable or expression evaluates to zero, or very close to zero.
	IsZero (see page 1236)	Indicates when a floating-point variable or expression evaluates to zero, or very close to zero.
	Ldexp (see page 1237)	Calculates X times (2 to the power of P).
	LnXP1 (see page 1237)	Returns the natural log of (X+1)
	Log10 (see page 1237)	Calculates log base 10.
	Log2 (see page 1238)	Calculates log base 2.
	LogN (see page 1238)	Calculates the log of X for a specified base.
	Max (see page 1239)	Returns the greater of two numeric values (Delphi only).
	Max (see page 1239)	Returns the greater of two numeric values (Delphi only).
	Max (see page 1239)	Returns the greater of two numeric values (Delphi only).
	Max (see page 1239)	Returns the greater of two numeric values (Delphi only).
	Max (see page 1239)	Returns the greater of two numeric values (Delphi only).
	MaxIntValue (see page 1239)	Returns the largest signed value in an integer array.
	MaxValue (see page 1240)	Returns the largest signed value in an array.
	Mean (see page 1240)	Returns the average of all values in an array.
	MeanAndStdDev (see page 1241)	Calculates the mean and standard deviation of array elements.
	Min (see page 1241)	Returns the lesser of two numeric values (Delphi only).
	Min (see page 1241)	Returns the lesser of two numeric values (Delphi only).
	Min (see page 1241)	Returns the lesser of two numeric values (Delphi only).
	Min (see page 1241)	Returns the lesser of two numeric values (Delphi only).
	Min (see page 1241)	Returns the lesser of two numeric values (Delphi only).
	MinIntValue (see page 1242)	Returns the smallest signed value in an integer array.
	MinValue (see page 1242)	Returns smallest signed value in an array.
	MomentSkewKurtosis (see page 1243)	Calculates the mean, variance, skew, and kurtosis.
	NetPresentValue (see page 1243)	Calculates the current value from an array of estimated cash flow values.
	Norm (see page 1244)	Returns the Euclidean 'L-2' norm.
	NumberOfPeriods (see page 1244)	Returns the number of payment periods for an investment.
	Payment (see page 1245)	Calculates a fully amortized payment.
	PeriodPayment (see page 1245)	Returns the principal amount from a full payment.
	Poly (see page 1245)	Evaluates a uniform polynomial of one variable at the value X.
	PopnStdDev (see page 1246)	Calculates the population standard deviation.
	PopnVariance (see page 1246)	Calculates the population variance.
	Power (see page 1247)	Raises Base to any power.
	Power (see page 1247)	Raises Base to any power.
	Power (see page 1247)	Raises Base to any power.
	PresentValue (see page 1247)	Calculates the present value of an investment.
	RadToCycle (see page 1248)	Converts radians to cycles.

	RadToDeg (see page 1248)	Converts radians to degrees.
	RadToGrad (see page 1248)	Converts radians to grads.
	RandG (see page 1249)	Generates random numbers with Gaussian distribution.
	RandomFrom (see page 1249)	Returns a randomly selected element from an array.
	RandomFrom (see page 1249)	Returns a randomly selected element from an array.
	RandomFrom (see page 1249)	Returns a randomly selected element from an array.
	RandomRange (see page 1250)	Returns a random integer from a specified range.
	RoundTo (see page 1250)	Rounds a floating-point value to a specified digit or power of ten using "Banker's rounding".
	SameValue (see page 1251)	Indicates whether two floating-point values are (approximately) equal.
	SameValue (see page 1251)	Indicates whether two floating-point values are (approximately) equal.
	SameValue (see page 1251)	Indicates whether two floating-point values are (approximately) equal.
	Sec (see page 1252)	Calculates the secant of an angle.
	Secant (see page 1252)	Calculates the secant of an angle.
	SecH (see page 1253)	Calculates the hyperbolic secant of an angle.
	SetExceptionMask (see page 1253)	Sets the exception mask on the FPU control word.
	SetPrecisionMode (see page 1254)	Sets the FPU precision control mode.
	SetRoundMode (see page 1254)	Sets the FPU rounding mode.
	Sign (see page 1255)	Indicates whether a numeric value is positive, negative, or zero.
	Sign (see page 1255)	Indicates whether a numeric value is positive, negative, or zero.
	Sign (see page 1255)	Indicates whether a numeric value is positive, negative, or zero.
	SimpleRoundTo (see page 1255)	Rounds a floating-point value to a specified digit or power of ten using asymmetric arithmetic rounding.
	SinCos (see page 1256)	Returns sine and cosine of an angle.
	Sinh (see page 1256)	Returns the hyperbolic sine of an angle.
	SLNDepreciation (see page 1257)	Returns the straight-line depreciation allowance of an asset.
	StdDev (see page 1257)	Returns the sample standard deviation for elements in an array.
	Sum (see page 1258)	Returns the sum of the elements in an array.
	SumInt (see page 1258)	Returns the sum of the elements in an integer array.
	SumOfSquares (see page 1259)	Returns the sum of the squared values from a data array.
	SumsAndSquares (see page 1259)	Returns the sum of the values and the sum of the squared values in an array.
	SYDDepreciation (see page 1260)	Calculates depreciation for an asset.
	Tan (see page 1260)	Returns the tangent of X.
	Tanh (see page 1261)	Returns the hyperbolic tangent of X.
	TotalVariance (see page 1261)	Returns the statistical variance from an array of values.
	Variance (see page 1262)	Calculates statistical sample variance from an array of data.

Structs, Records, Enums

	Name	Description
	TFPUException (see page 1263)	TFPUException defines TFPUExceptionMask (see page 1267) values.
	TFPUPrecisionMode (see page 1264)	TFPUPrecisionMode defines how the FPU handles precision.
	TFPURoundingMode (see page 1265)	TFPURoundingMode defines how the FPU handles rounding issues.
	TPaymentTime (see page 1266)	TPaymentTime is an enumerated type used as a parameter in financial functions.

Types

Name	Description
TFPUExceptionMask (see page 1267)	TFPUExceptionMask defines a set of TFPUException (see page 1263) values.
TRoundToEXRangeExtended (see page 1268)	This is type Math.TRoundToEXRangeExtended.
TRoundToRange (see page 1268)	TRoundToRange defines the RoundTo (see page 1250) rounding limits.
TValueSign (see page 1268)	TValueSign defines the Sign (see page 1255) returned value.

32.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	EInvalidArgument (see page 1209)	EInvalidArgument is the exception class for out-of-range parameters in specialized math and accounting functions.

32.1.1 EInvalidArgument Class

EInvalidArgument is the exception class for out-of-range parameters in specialized math and accounting functions.

Class Hierarchy

Pascal

```
EInvalidArgument = class(EMathError);
```

C++

```
class EInvalidArgument : public EMathError;
```

File

Math

Description

EInvalidArgument is raised by some functions in the Math (see page 1205) unit (mostly financial functions) when one of the parameters is out of range or makes the calculation impossible.

Note: On Windows, because EInvalidArgument descends (indirectly) from EExternal, it saves the Windows exception record (if there is one) in its ExceptionRecord data structure.

See Also

EExternal

DoubleDecliningBalance (see page 1226)

FutureValue (see page 1228)

InterestPayment (see page 1233)

NetPresentValue (see page 1243)

PeriodPayment (see page 1245)

Power (see page 1247)

SLNDepreciation (see page 1257)

SYDDepreciation (see page 1260)

EInvalidOp

EOverflow

EUnderflow

EZeroDivide

32.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	ArcCos (see page 1213)	Calculates the inverse cosine of a given number.
	ArcCos (see page 1213)	Calculates the inverse cosine of a given number.
	ArcCos (see page 1213)	Calculates the inverse cosine of a given number.
	ArcCosh (see page 1214)	Calculates the inverse hyperbolic cosine of a given number.
	ArcCot (see page 1214)	Calculates the inverse cotangent of a given number.
	ArcCotH (see page 1215)	Calculates the inverse hyperbolic cotangent of a given number.
	ArcCsc (see page 1215)	Calculates the inverse cosecant of a given number.
	ArcCscH (see page 1216)	Calculates the inverse hyperbolic cosecant of a given number.
	ArcSec (see page 1216)	Calculates the inverse secant of a given number.
	ArcSecH (see page 1216)	Calculates the inverse hyperbolic secant of a given number.
	ArcSin (see page 1217)	Calculates the inverse sine of a given number.
	ArcSin (see page 1217)	Calculates the inverse sine of a given number.
	ArcSin (see page 1217)	Calculates the inverse sine of a given number.
	ArcSinh (see page 1217)	Calculates the inverse hyperbolic sine of a given number.
	ArcTan2 (see page 1218)	Calculates the arctangent angle and quadrant of a given number.
	ArcTanh (see page 1218)	Calculates the inverse hyperbolic tangent of a given number.
	Ceil (see page 1219)	Rounds variables up toward positive infinity.
	ClearExceptions (see page 1219)	Clears exception bits in FPU status word.
	CompareValue (see page 1220)	Returns the relationship between two numeric values.
	CompareValue (see page 1220)	Returns the relationship between two numeric values.
	CompareValue (see page 1220)	Returns the relationship between two numeric values.
	CompareValue (see page 1220)	Returns the relationship between two numeric values.
	CompareValue (see page 1220)	Returns the relationship between two numeric values.
	Cosecant (see page 1220)	Returns the cosecant of an angle.
	Cosh (see page 1221)	Calculates the hyperbolic cosine of an angle.
	Cot (see page 1221)	Calculates the cotangent of an angle.
	Cotan (see page 1222)	Calculates the cotangent of an angle.
	CotH (see page 1222)	Calculates the hyperbolic cotangent of an angle.
	Csc (see page 1223)	Returns the cosecant of an angle.
	CscH (see page 1223)	Returns the hyperbolic cosecant of an angle.
	CycleToDeg (see page 1224)	Converts an angle measurement from cycles to degrees.
	CycleToGrad (see page 1224)	Converts an angle measurement from cycles to grads.
	CycleToRad (see page 1224)	Converts an angle measurement from cycles to radians.
	DegToCycle (see page 1225)	Returns the value of a degree measurement expressed in cycles.
	DegToGrad (see page 1225)	Returns the value of a degree measurement expressed in grads.
	DegToRad (see page 1226)	Returns the value of a degree measurement expressed in radians.
	DivMod (see page 1226)	Returns the result of an integer division, including the remainder.
	DoubleDecliningBalance (see page 1226)	Calculates the depreciation of an asset using the double-declining balance method.
	EnsureRange (see page 1227)	Returns the closest value to a specified value within a specified range.
	EnsureRange (see page 1227)	Returns the closest value to a specified value within a specified range.
	EnsureRange (see page 1227)	Returns the closest value to a specified value within a specified range.
	Floor (see page 1227)	Rounds variables toward negative infinity.
	Frexp (see page 1228)	Separates the Mantissa and Exponent of X
	FutureValue (see page 1228)	Calculates the future value of an investment.

	GetExceptionMask (see page 1229)	Returns the exception mask from the FPU control word.
	GetPrecisionMode (see page 1229)	Returns the FPU precision control mode.
	GetRoundMode (see page 1230)	Returns the FPU rounding mode.
	GradToCycle (see page 1231)	Converts grad measurements to cycles.
	GradToDeg (see page 1231)	Converts grad measurements to degrees.
	GradToRad (see page 1231)	Converts grad measurements to radians.
	Hypot (see page 1232)	Calculates the length of the hypotenuse.
	IfThen (see page 1232)	Conditionally returns one of two specified values.
	IfThen (see page 1232)	Conditionally returns one of two specified values.
	IfThen (see page 1232)	Conditionally returns one of two specified values.
	InRange (see page 1232)	Indicates whether a value falls within a specified range.
	InRange (see page 1232)	Indicates whether a value falls within a specified range.
	InRange (see page 1232)	Indicates whether a value falls within a specified range.
	InterestPayment (see page 1233)	Calculates the interest portion of a loan payment.
	InterestRate (see page 1234)	Returns the interest rate required to increase PresentValue (see page 1247) to FutureValue (see page 1228).
	InternalRateOfReturn (see page 1234)	Calculates the internal rate of return for an investment.
	IntPower (see page 1235)	Calculates the integral power of a base value.
	IsInfinite (see page 1235)	Indicates when a variable or expression represents an infinite value.
	IsNan (see page 1236)	Indicates when a variable or expression does not evaluate to a numeric value.
	IsNan (see page 1236)	Indicates when a variable or expression does not evaluate to a numeric value.
	IsNan (see page 1236)	Indicates when a variable or expression does not evaluate to a numeric value.
	IsZero (see page 1236)	Indicates when a floating-point variable or expression evaluates to zero, or very close to zero.
	IsZero (see page 1236)	Indicates when a floating-point variable or expression evaluates to zero, or very close to zero.
	IsZero (see page 1236)	Indicates when a floating-point variable or expression evaluates to zero, or very close to zero.
	Ldexp (see page 1237)	Calculates X times (2 to the power of P).
	LnXP1 (see page 1237)	Returns the natural log of (X+1)
	Log10 (see page 1237)	Calculates log base 10.
	Log2 (see page 1238)	Calculates log base 2.
	LogN (see page 1238)	Calculates the log of X for a specified base.
	Max (see page 1239)	Returns the greater of two numeric values (Delphi only).
	Max (see page 1239)	Returns the greater of two numeric values (Delphi only).
	Max (see page 1239)	Returns the greater of two numeric values (Delphi only).
	Max (see page 1239)	Returns the greater of two numeric values (Delphi only).
	Max (see page 1239)	Returns the greater of two numeric values (Delphi only).
	MaxIntValue (see page 1239)	Returns the largest signed value in an integer array.
	MaxValue (see page 1240)	Returns the largest signed value in an array.
	Mean (see page 1240)	Returns the average of all values in an array.
	MeanAndStdDev (see page 1241)	Calculates the mean and standard deviation of array elements.
	Min (see page 1241)	Returns the lesser of two numeric values (Delphi only).
	Min (see page 1241)	Returns the lesser of two numeric values (Delphi only).
	Min (see page 1241)	Returns the lesser of two numeric values (Delphi only).
	Min (see page 1241)	Returns the lesser of two numeric values (Delphi only).
	Min (see page 1241)	Returns the lesser of two numeric values (Delphi only).
	MinIntValue (see page 1242)	Returns the smallest signed value in an integer array.
	MinValue (see page 1242)	Returns smallest signed value in an array.
	MomentSkewKurtosis (see page 1243)	Calculates the mean, variance, skew, and kurtosis.
	NetPresentValue (see page 1243)	Calculates the current value from an array of estimated cash flow values.
	Norm (see page 1244)	Returns the Euclidean 'L-2' norm.
	NumberOfPeriods (see page 1244)	Returns the number of payment periods for an investment.
	Payment (see page 1245)	Calculates a fully amortized payment.
	PeriodPayment (see page 1245)	Returns the principal amount from a full payment.
	Poly (see page 1245)	Evaluates a uniform polynomial of one variable at the value X.

	PopnStdDev (see page 1246)	Calculates the population standard deviation.
	PopnVariance (see page 1246)	Calculates the population variance.
	Power (see page 1247)	Raises Base to any power.
	Power (see page 1247)	Raises Base to any power.
	Power (see page 1247)	Raises Base to any power.
	PresentValue (see page 1247)	Calculates the present value of an investment.
	RadToCycle (see page 1248)	Converts radians to cycles.
	RadToDeg (see page 1248)	Converts radians to degrees.
	RadToGrad (see page 1248)	Converts radians to grads.
	RandG (see page 1249)	Generates random numbers with Gaussian distribution.
	RandomFrom (see page 1249)	Returns a randomly selected element from an array.
	RandomFrom (see page 1249)	Returns a randomly selected element from an array.
	RandomFrom (see page 1249)	Returns a randomly selected element from an array.
	RandomRange (see page 1250)	Returns a random integer from a specified range.
	RoundTo (see page 1250)	Rounds a floating-point value to a specified digit or power of ten using "Banker's rounding".
	SameValue (see page 1251)	Indicates whether two floating-point values are (approximately) equal.
	SameValue (see page 1251)	Indicates whether two floating-point values are (approximately) equal.
	SameValue (see page 1251)	Indicates whether two floating-point values are (approximately) equal.
	Sec (see page 1252)	Calculates the secant of an angle.
	Secant (see page 1252)	Calculates the secant of an angle.
	SecH (see page 1253)	Calculates the hyperbolic secant of an angle.
	SetExceptionMask (see page 1253)	Sets the exception mask on the FPU control word.
	SetPrecisionMode (see page 1254)	Sets the FPU precision control mode.
	SetRoundMode (see page 1254)	Sets the FPU rounding mode.
	Sign (see page 1255)	Indicates whether a numeric value is positive, negative, or zero.
	Sign (see page 1255)	Indicates whether a numeric value is positive, negative, or zero.
	Sign (see page 1255)	Indicates whether a numeric value is positive, negative, or zero.
	SimpleRoundTo (see page 1255)	Rounds a floating-point value to a specified digit or power of ten using asymmetric arithmetic rounding.
	SinCos (see page 1256)	Returns sine and cosine of an angle.
	Sinh (see page 1256)	Returns the hyperbolic sine of an angle.
	SLNDepreciation (see page 1257)	Returns the straight-line depreciation allowance of an asset.
	StdDev (see page 1257)	Returns the sample standard deviation for elements in an array.
	Sum (see page 1258)	Returns the sum of the elements in an array.
	SumInt (see page 1258)	Returns the sum of the elements in an integer array.
	SumOfSquares (see page 1259)	Returns the sum of the squared values from a data array.
	SumsAndSquares (see page 1259)	Returns the sum of the values and the sum of the squared values in an array.
	SYDDepreciation (see page 1260)	Calculates depreciation for an asset.
	Tan (see page 1260)	Returns the tangent of X.
	Tanh (see page 1261)	Returns the hyperbolic tangent of X.
	TotalVariance (see page 1261)	Returns the statistical variance from an array of values.
	Variance (see page 1262)	Calculates statistical sample variance from an array of data.

32.2.1 Math.ArcCos Function

Calculates the inverse cosine of a given number.

Pascal

```
function ArcCos(const X: Extended): Extended; overload;
function ArcCos(const X: Double): Double; overload;
function ArcCos(const X: Single): Single; overload;
```

C++

```
Extended ArcCos(const Extended X);
Double ArcCos(const Double X);
Single ArcCos(const Single X);
```

File

Math

Description

ArcCos returns the inverse cosine of X.

X must be between -1 and 1.

The return value is in the range [0..Pi], in radians.

See Also

Cos

ArcCosh (see page 1214)

ArcSin (see page 1217)

32.2.2 Math.ArcCosh Function

Calculates the inverse hyperbolic cosine of a given number.

Pascal

```
function ArcCosh(const X: Extended): Extended;
```

C++

```
Extended ArcCosh(const Extended X);
```

File

Math

Description

ArcCosh returns the inverse hyperbolic cosine of X. The value of X must be greater than or equal to 1.

See Also

Cosh (see page 1221)

ArcCos (see page 1213)

32.2.3 Math.ArcCot Function

Calculates the inverse cotangent of a given number.

Pascal

```
function ArcCot(const X: Extended): Extended;
```

C++

```
Extended ArcCot(const Extended X);
```

File

Math

Description

ArcCot returns the inverse cotangent of X.

See Also

Cot (see page 1221)

ArcCotH (see page 1215)

32.2.4 Math.ArcCotH Function

Calculates the inverse hyperbolic cotangent of a given number.

Pascal

```
function ArcCotH(const X: Extended): Extended;
```

C++

```
Extended ArcCotH(const Extended X);
```

File

Math

Description

ArcCot (see page 1214) returns the inverse hyperbolic cotangent of X.

See Also

Cot (see page 1221)

32.2.5 Math.ArcCsc Function

Calculates the inverse cosecant of a given number.

Pascal

```
function ArcCsc(const X: Extended): Extended;
```

C++

```
Extended ArcCsc(const Extended X);
```

File

Math

Description

ArcCsc returns the inverse cosecant of X.

See Also

Csc (see page 1223)

ArcCscH (see page 1216)

32.2.6 Math.ArcCscH Function

Calculates the inverse hyperbolic cosecant of a given number.

Pascal

```
function ArcCscH(const X: Extended): Extended;
```

C++

```
Extended ArcCscH(const Extended X);
```

File

Math

Description

ArcCsc (see page 1215) returns the inverse hyperbolic cosecant of X.

See Also

Csc (see page 1223)

CscH (see page 1223)

32.2.7 Math.ArcSec Function

Calculates the inverse secant of a given number.

Pascal

```
function ArcSec(const X: Extended): Extended;
```

C++

```
Extended ArcSec(const Extended X);
```

File

Math

Description

ArcSec returns the inverse secant of X.

See Also

Sec (see page 1252)

ArcSecH (see page 1216)

32.2.8 Math.ArcSecH Function

Calculates the inverse hyperbolic secant of a given number.

Pascal

```
function ArcSecH(const X: Extended): Extended;
```

C++

```
Extended ArcSecH(const Extended X);
```

File

Math

Description

ArcSec (see page 1216) returns the inverse hyperbolic secant of X.

See Also

Sec (see page 1252)

SecH (see page 1253)

32.2.9 Math.ArcSin Function

Calculates the inverse sine of a given number.

Pascal

```
function ArcSin(const X: Extended): Extended; overload;
function ArcSin(const X: Double): Double; overload;
function ArcSin(const X: Single): Single; overload;
```

C++

```
Extended ArcSin(const Extended X);
Double ArcSin(const Double X);
Single ArcSin(const Single X);
```

File

Math

Description

ArcSin returns the inverse sine of X. X must be between -1 and 1. The return value will be in the range [-Pi/2..Pi/2], in radians.

See Also

Sin

ArcSinh (see page 1217)

ArcCos (see page 1213)

32.2.10 Math.ArcSinh Function

Calculates the inverse hyperbolic sine of a given number.

Pascal

```
function ArcSinh(const X: Extended): Extended;
```

C++

```
Extended ArcSinh(const Extended X);
```

File

Math

Description

ArcSinh returns the inverse hyperbolic sine of X.

See Also

Sinh (see page 1256)

32.2.11 Math.ArcTan2 Function

Calculates the arctangent angle and quadrant of a given number.

Pascal

```
function ArcTan2(const Y: Extended; const X: Extended): Extended;
```

C++

```
Extended ArcTan2(const Extended Y, const Extended X);
```

File

Math

Description

ArcTan2 calculates ArcTan(Y/X), and returns an angle in the correct quadrant. The values of X and Y must be between –2^64 and 2^64. In addition, the value of X can't be 0. The return value will fall in the range from -Pi to Pi radians.

32.2.12 Math.ArcTanh Function

Calculates the inverse hyperbolic tangent of a given number.

Pascal

```
function ArcTanh(const X: Extended): Extended;
```

C++

```
Extended ArcTanh(const Extended X);
```

File

Math

Description

ArcTanh returns the inverse hyperbolic tangent of X. The value of X must be between -1 and 1 (inclusive).

See Also

Tanh (see page 1261)

32.2.13 Math.Ceil Function

Rounds variables up toward positive infinity.

Pascal

```
function Ceil(const X: Extended): Integer;
```

C++

```
int Ceil(const Extended X);
```

File

Math

Description

Call Ceil to obtain the lowest integer greater than or equal to X. The absolute value of X must be less than MaxInt. For example:

Ceil(-2.8) = -2

Ceil(2.8) = 3

Ceil(-1.0) = -1

See Also

Trunc

Floor (see page 1227)

32.2.14 Math.ClearExceptions Function

Clears exception bits in FPU status word.

Pascal

```
procedure ClearExceptions(RaisePending: Boolean = True);
```

C++

```
ClearExceptions(Boolean RaisePending = True);
```

File

Math

Description

ClearExceptions clears the exception bits in the FPU status word. If RaisePending is true, any pending floating point exceptions cause an error to be raised.

See Also

GetExceptionMask (see page 1229)

SetExceptionMask (see page 1253)

Set8087CW

32.2.15 Math.CompareValue Function

Returns the relationship between two numeric values.

Pascal

```
function CompareValue(const A: Extended; const B: Extended; Epsilon: Extended = 0):
TValueRelationship; overload;
function CompareValue(const A: Double; const B: Double; Epsilon: Double = 0):
TValueRelationship; overload;
function CompareValue(const A: Single; const B: Single; Epsilon: Single = 0):
TValueRelationship; overload;
function CompareValue(const A: Integer; const B: Integer): TValueRelationship; overload;
function CompareValue(const A: Int64; const B: Int64): TValueRelationship; overload;
```

C++

```
TValueRelationship CompareValue(const Extended A, const Extended B, Extended Epsilon = 0);
TValueRelationship CompareValue(const Double A, const Double B, Double Epsilon = 0);
TValueRelationship CompareValue(const Single A, const Single B, Single Epsilon = 0);
TValueRelationship CompareValue(const int A, const int B);
TValueRelationship CompareValue(const Int64 A, const Int64 B);
```

File

Math

Description

Call CompareValue to determine the relationship between two numeric values. When comparing floating-point values, CompareValue lets you specify a maximum difference to use when comparing values, so that they are considered the same if they are within that amount.

A and B are the values to compare.

Epsilon is the maximum amount by which A and B can differ and still be considered the same value.

CompareValue returns

LessThanValue if A is less than B (by more than Epsilon if A and B are floating-point numbers).

EqualsValue if A is equivalent to B (the same, or within Epsilon if A and B are floating-point numbers).

GreaterThanValue if A is larger than B (by more than Epsilon if A and B are floating-point numbers).

See Also

SameValue (see page 1251)

VarCompareValue

32.2.16 Math.Cosecant Function

Returns the cosecant of an angle.

Pascal

```
function Cosecant(const X: Extended): Extended;
```

C++

```
Extended Cosecant(const Extended X);
```

File

Math

Description

Use the Cosecant to calculate the cosecant of X, where X is an angle in radians. The cosecant is calculated as 1/ Sin(X).

See Also

ArcCsc (see page 1215)

CscH (see page 1223)

Secant (see page 1252)

Sin

32.2.17 Math.Cosh Function

Calculates the hyperbolic cosine of an angle.

Pascal

```
function Cosh(const X: Extended): Extended;
```

C++

```
Extended Cosh(const Extended X);
```

File

Math

Description

Use the Cosh to calculate the hyperbolic cosine of X.

See Also

Cos

ArcCosh (see page 1214)

Sinh (see page 1256)

32.2.18 Math.Cot Function

Calculates the cotangent of an angle.

Pascal

```
function Cot(const X: Extended): Extended;
```

C++

```
Extended Cot(const Extended X);
```

File

Math

Description

Call Cot to obtain the cotangent of X. The cotangent is calculated using the formula 1 / Tan (see page 1260)(X).

See Also

Tan (see page 1260)

Sin

Cos

ArcTan

32.2.19 Math.Cotan Function

Calculates the cotangent of an angle.

Pascal

```
function Cotan(const X: Extended): Extended;
```

C++

```
Extended Cotan(const Extended X);
```

File

Math

Description

Call Cotan to obtain the cotangent of X. The cotangent is calculated using the formula

1 / Tan (see page 1260)(X)

Note: Do not call Cotan with X = 0.

See Also

Tan (see page 1260)

Sin

Cos

ArcTan

32.2.20 Math.CotH Function

Calculates the hyperbolic cotangent of an angle.

Pascal

```
function CotH(const X: Extended): Extended;
```

C++

```
Extended CotH(const Extended X);
```

File

Math

Description

Call CotH to obtain the hyperbolic cotangent of X, where X is an angle in Radians.

See Also

Cot (see page 1221)

ArcCot (see page 1214)

32.2.21 Math.Csc Function

Returns the cosecant of an angle.

Pascal

```
function Csc(const X: Extended): Extended;
```

C++

```
Extended Csc(const Extended X);
```

File

Math

Description

Use the Csc to calculate the cosecant of X, where X is an angle in radians.

See Also

ArcCsc (see page 1215)

CscH (see page 1223)

Sec (see page 1252)

Sin

32.2.22 Math.CscH Function

Returns the hyperbolic cosecant of an angle.

Pascal

```
function CscH(const X: Extended): Extended;
```

C++

```
Extended CscH(const Extended X);
```

File

Math

Description

Use the CscH to calculate the hyperbolic cosecant of X, where X is an angle in radians.

See Also

ArcCscH (see page 1216)

Csc (see page 1223)

SecH (see page 1253)

32.2.23 Math.CycleToDeg Function

Converts an angle measurement from cycles to degrees.

Pascal

```
function CycleToDeg(const Cycles: Extended): Extended;
```

C++

```
Extended CycleToDeg(const Extended Cycles);
```

File

Math

Description

CycleToDeg converts angles measured in cycles into degrees, where

degrees = cycles * 360.

32.2.24 Math.CycleToGrad Function

Converts an angle measurement from cycles to grads.

Pascal

```
function CycleToGrad(const Cycles: Extended): Extended;
```

C++

```
Extended CycleToGrad(const Extended Cycles);
```

File

Math

Description

CycleToGrad converts angles measured in cycles into grads.

32.2.25 Math.CycleToRad Function

Converts an angle measurement from cycles to radians.

Pascal

```
function CycleToRad(const Cycles: Extended): Extended;
```

C++

```
Extended CycleToRad(const Extended Cycles);
```

File

Math

Description

CycleToRad converts angles measured in cycles into radians, where

radians = 2pi * cycles.

32.2.26 Math.DegToCycle Function

Returns the value of a degree measurement expressed in cycles.

Pascal

```
function DegToCycle(const Degrees: Extended): Extended;
```

C++

```
Extended DegToCycle(const Extended Degrees);
```

File

Math

Description

Use DegToCycle to convert angles expressed in degrees to the corresponding value in cycles.

32.2.27 Math.DegToGrad Function

Returns the value of a degree measurement expressed in grads.

Pascal

```
function DegToGrad(const Degrees: Extended): Extended;
```

C++

```
Extended DegToGrad(const Extended Degrees);
```

File

Math

Description

Use DegToGrad to convert angles expressed in degrees to the corresponding value in grads.

32.2.28 Math.DegToRad Function

Returns the value of a degree measurement expressed in radians.

Pascal

```
function DegToRad(const Degrees: Extended): Extended;
```

C++

```
Extended DegToRad(const Extended Degrees);
```

File

Math

Description

Use DegToRad to convert angles expressed in degrees to the corresponding value in radians, where

radians = degrees(pi/180).

32.2.29 Math.DivMod Function

Returns the result of an integer division, including the remainder.

Pascal

```
procedure DivMod(Dividend: Cardinal; Divisor: Word; var Result: Word; var Remainder: Word);
```

C++

```
DivMod(unsigned Dividend, Word Divisor, Word Result, Word Remainder);
```

File

Math

Description

Call DivMod to perform 16-bit integer division and fetch the remainder as well, all in one operation.

Dividend is the integer into which you are dividing.

Divisor is the value by which to divide Dividend.

Result returns the result of the integer division.

Remainder returns the remainder (the difference between Result * Divisor and Dividend).

32.2.30 Math.DoubleDecliningBalance Function

Calculates the depreciation of an asset using the double-declining balance method.

Pascal

```
function DoubleDecliningBalance(const Cost: Extended; const Salvage: Extended; Life: Integer;
Period: Integer): Extended;
```

C++

```
Extended DoubleDecliningBalance(const Extended Cost, const Extended Salvage, int Life, int
Period);
```

File

Math

Description

DoubleDecliningBalance determines accelerated depreciation values for an asset, given the initial cost, life expectancy, end value, and depreciation period.

32.2.31 Math.EnsureRange Function

Returns the closest value to a specified value within a specified range.

Pascal

```
function EnsureRange(const AValue: Integer; const AMin: Integer; const AMax: Integer):
Integer; overload;
function EnsureRange(const AValue: Int64; const AMin: Int64; const AMax: Int64): Int64;
overload;
function EnsureRange(const AValue: Double; const AMin: Double; const AMax: Double): Double;
overload;
```

C++

```
int EnsureRange(const int AValue, const int AMin, const int AMax);
Int64 EnsureRange(const Int64 AValue, const Int64 AMin, const Int64 AMax);
Double EnsureRange(const Double AValue, const Double AMin, const Double AMax);
```

File

Math

Description

EnsureRange returns a value that is guaranteed to be within a specified range.

AValue is the value you want as long as it is in range. If AValue is in range, EnsureRange returns AValue.

AMin is the minimum acceptable value. If AValue is less than AMin, EnsureRange returns AMin.

AMax is the maximum acceptable value. If AValue is greater than AMax, EnsureRange returns AMax.

See Also

VarEnsureRange

InRange (see page 1232)

32.2.32 Math.Floor Function

Rounds variables toward negative infinity.

Pascal

```
function Floor(const X: Extended): Integer;
```

C++

```
int Floor(const Extended X);
```

File

Math

Description

Call Floor to obtain the highest integer less than or equal to X. For example:

Floor(-2.8) = -3

Floor(2.8) = 2

Floor(-1.0) = -1

Note: The absolute value of X must be less than MaxInt.

See Also

Trunc

Ceil (see page 1219)

32.2.33 Math.Frexp Function

Separates the Mantissa and Exponent of X

Pascal

```
procedure Frexp(const X: Extended; var Mantissa: Extended; var Exponent: Integer);
```

C++

```
Frexp(const Extended X, Extended Mantissa, int Exponent);
```

File

Math

Description

Frexp returns the mantissa of X as Mantissa and the exponent as Exponent.

32.2.34 Math.FutureValue Function

Calculates the future value of an investment.

Pascal

```
function FutureValue(const Rate: Extended; NPeriods: Integer; const Payment: Extended; const
PresentValue: Extended; PaymentTime: TPaymentTime): Extended;
```

C++

```
Extended FutureValue(const Extended Rate, int NPeriods, const Extended Payment, const Extended
PresentValue, TPaymentTime PaymentTime);
```

File

Math

Description

FutureValue returns the future value of an investment of PresentValue (see page 1247) where Payment (see page 1245) is invested for NPeriods at the rate of Rate per period. The PaymentTime parameter indicates whether the investment is an ordinary annuity or an annuity due (enter ptEndOfPeriod if payments are at the end of each period, ptStartOfPeriod if they are at the beginning).

32.2.35 Math.GetExceptionMask Function

Returns the exception mask from the FPU control word.

Pascal

```
function GetExceptionMask: TFPUExceptionMask;
```

C++

```
TFPUExceptionMask GetExceptionMask();
```

File

Math

Description

Call GetExceptionMask to ascertain the types of exceptions the FPU (floating-point unit) generates. Any element set in this mask prevents the FPU from raising the corresponding type of exception. Instead, the FPU returns its best attempt at a value – often NaN (see page 1274), Infinity (see page 1270), or NegInfinity (see page 1275). The value depends on the operation and the current rounding mode.

See Also

Set8087CW

SetExceptionMask (see page 1253)

NaN (see page 1274)

Infinity (see page 1270)

NegInfinity (see page 1275)

IsInfinite (see page 1235)

IsNan (see page 1236)

32.2.36 Math.GetPrecisionMode Function

Returns the FPU precision control mode.

Pascal

```
function GetPrecisionMode: TFPUPrecisionMode;
```

C++

```
TFPUPrecisionMode GetPrecisionMode();
```

File

Math

Description

Call GetPrecisionMode to ascertain the level of precision that the FPU (floating-point unit) uses for floating-point calculations. The precision control mode can be any of the following values:

Value	Meaning
pmSingle	single precision
pmReserved	not used
pmDouble	double precision
pmExtended	extended precision

See Also

Set8087CW

SetPrecisionMode (see page 1254)

32.2.37 Math.GetRoundMode Function

Returns the FPU rounding mode.

Pascal

```
function GetRoundMode: TFPURoundingMode;
```

C++

```
TFPURoundingMode GetRoundMode();
```

File

Math

Description

Call GetRoundingMode to ascertain how the FPU handles rounding issues. The rounding mode can be any of the following values:

Value	Meaning
rmNearest	Rounds to the closest value.
rmDown	Rounds toward negative infinity.
rmUp	Rounds toward positive infinity.
rmTruncate	Truncates the value, rounding positive numbers down and negative numbers up.

See Also

Set8087CW

SetRoundMode (see page 1254)

32.2.38 Math.GradToCycle Function

Converts grad measurements to cycles.

Pascal

```
function GradToCycle(const Grads: Extended): Extended;
```

C++

```
Extended GradToCycle(const Extended Grads);
```

File

Math

Description

GradToCycle converts angles measured in grads into cycles.

32.2.39 Math.GradToDeg Function

Converts grad measurements to degrees.

Pascal

```
function GradToDeg(const Grads: Extended): Extended;
```

C++

```
Extended GradToDeg(const Extended Grads);
```

File

Math

Description

GradToDeg converts angles measured in grads into degrees.

32.2.40 Math.GradToRad Function

Converts grad measurements to radians.

Pascal

```
function GradToRad(const Grads: Extended): Extended;
```

C++

```
Extended GradToRad(const Extended Grads);
```

File

Math

Description

GradToRad converts angles measured in grads into radians, where

radians = grads(pi/200).

32.2.41 Math.Hypot Function

Calculates the length of the hypotenuse.

Pascal

```
function Hypot(const X: Extended; const Y: Extended): Extended;
```

C++

```
Extended Hypot(const Extended X, const Extended Y);
```

File

Math

Description

Hypot returns the length of the hypotenuse of a right triangle. Specify the lengths of the sides adjacent to the right angle in X and Y. Hypot uses the formula

Sqrt(X**2 + Y**2)

32.2.42 Math.IfThen Function

Conditionally returns one of two specified values.

Pascal

```
function IfThen(AValue: Boolean; const ATrue: Integer; const AFalse: Integer = 0): Integer;
overload;
function IfThen(AValue: Boolean; const ATrue: Int64; const AFalse: Int64 = 0): Int64; overload;
function IfThen(AValue: Boolean; const ATrue: Double; const AFalse: Double = 0.0): Double;
overload;
```

C++

```
int IfThen(Boolean AValue, const int ATrue, const int AFalse = 0);
Int64 IfThen(Boolean AValue, const Int64 ATrue, const Int64 AFalse = 0);
Double IfThen(Boolean AValue, const Double ATrue, const Double AFalse = 0.0);
```

File

Math

Description

IfThen checks the expression passed as AValue and returns ATrue if it evaluates to true, or AFalse if it evaluates to false. In Delphi, if the AFalse parameter is omitted, IfThen returns 0 or an empty string when AValue evaluates to False.

32.2.43 Math.InRange Function

Indicates whether a value falls within a specified range.

Pascal

```
function InRange(const AValue: Integer; const AMin: Integer; const AMax: Integer): Boolean;
overload;
function InRange(const AValue: Int64; const AMin: Int64; const AMax: Int64): Boolean; overload;
function InRange(const AValue: Double; const AMin: Double; const AMax: Double): Boolean;
overload;
```

C++

```
Boolean InRange(const int AValue, const int AMin, const int AMax);
Boolean InRange(const Int64 AValue, const Int64 AMin, const Int64 AMax);
Boolean InRange(const Double AValue, const Double AMin, const Double AMax);
```

File

Math

Description

InRange returns true if AValue is greater than or equal to AMin and less than or Equal to AMax. It returns false if AValue is less than AMin or greater than AMax.

See Also

VarInRange

EnsureRange (see page 1227)

32.2.44 Math.InterestPayment Function

Calculates the interest portion of a loan payment.

Pascal

```
function InterestPayment(const Rate: Extended; Period: Integer; NPeriods: Integer; const
PresentValue: Extended; const FutureValue: Extended; PaymentTime: TPaymentTime): Extended;
```

C++

```
Extended InterestPayment(const Extended Rate, int Period, int NPeriods, const Extended
PresentValue, const Extended FutureValue, TPaymentTime PaymentTime);
```

File

Math

Description

Call InterestPayment to calculate the portion of a loan payment that reflects the interest.

Rate represents the fixed periodic interest rate.

Period identifies the payment period.

NPeriods is the number of periods of the loan.

PresentValue (see page 1247) represents the amount borrowed (the principal).

FutureValue (see page 1228) is the future value of the investment.

PaymentTime indicates whether the cash flows occur at the beginning or end of the period.

32.2.45 Math.InterestRate Function

Returns the interest rate required to increase PresentValue (see page 1247) to FutureValue (see page 1228).

Pascal

```
function InterestRate(NPeriods: Integer; const Payment: Extended; const PresentValue:
Extended; const FutureValue: Extended; PaymentTime: TPaymentTime): Extended;
```

C++

```
Extended InterestRate(int NPeriods, const Extended Payment, const Extended PresentValue, const
Extended FutureValue, TPaymentTime PaymentTime);
```

File

Math

Description

InterestRate calculates the interest rate required in order for an investment of PresentValue (see page 1247), with periodic payments of Payment (see page 1245), to be worth FutureValue (see page 1228) within NPeriods compounding periods. If NPeriods represents years, an annual interest rate results; if NPeriods represents months, a monthly interest rate results, and so on. The PaymentTime parameter indicates whether the cash flows occur at the beginning or end of the period.

32.2.46 Math.InternalRateOfReturn Function

Calculates the internal rate of return for an investment.

Pascal

```
function InternalRateOfReturn(const Guess: Extended; const CashFlows: array of Double):
Extended;
```

C++

```
Extended InternalRateOfReturn(const Extended Guess, const array of Double CashFlows);
```

File

Math

Description

InternalRateOfReturn determines the internal rate of return on an investment. It references an array that contains cash flow information and uses the supplied internal rate of return estimate to calculate results.

Before using this function, define an array containing expected cash flow amounts over a period of time. It is assumed that the amounts are received at regular intervals. Negative amounts are interpreted as cash outflows, and positive amounts as inflows. The first amount must be a negative number, to reflect the initial investment. The following amounts can all be the same for each time period, or they can be different (including a mixture of negatives, positives, or zeros).

Specify the estimated rate of return as the Guess parameter. Pass the array of expected cash flow amounts as the CashFlows parameter. In C++, the CashFlows_Size parameter gives the index of the last entry in CashFlows (one less than the number of entries).

32.2.47 Math.IntPower Function

Calculates the integral power of a base value.

Pascal

```
function IntPower(const Base: Extended; const Exponent: Integer): Extended;
```

C++

```
Extended IntPower(const Extended Base, const int Exponent);
```

File

Math

Description

IntPower raises Base to the power specified by Exponent.

See Also

Power (see page 1247)

32.2.48 Math.IsInfinite Function

Indicates when a variable or expression represents an infinite value.

Pascal

```
function IsInfinite(const AValue: Double): Boolean;
```

C++

```
Boolean IsInfinite(const Double AValue);
```

File

Math

Description

Use IsInfinite to test whether the value specified by AValue represents positive or negative infinity. Use the Sign (see page 1255) function to distinguish positive infinity from negative infinity.

See Also

GetExceptionMask (see page 1229)

NegInfinity (see page 1275)

IsNan (see page 1236)

Infinity (see page 1270)

IsZero (see page 1236)

Sign (see page 1255)

32.2.49 Math.IsNan Function

Indicates when a variable or expression does not evaluate to a numeric value.

Pascal

```
function IsNan(const AValue: Double): Boolean; overload;
function IsNan(const AValue: Single): Boolean; overload;
function IsNan(const AValue: Extended): Boolean; overload;
```

C++

```
Boolean IsNan(const Double AValue);
Boolean IsNan(const Single AValue);
Boolean IsNan(const Extended AValue);
```

File

Math

Description

Use IsNan to test whether the value specified by AValue represents NaN (see page 1274) (not a number). Note that NaN (see page 1274) is distinct from positive and negative infinity, which can be detected using the IsInfinite (see page 1235) function.

See Also

GetExceptionMask (see page 1229)

NaN (see page 1274)

IsInfinite (see page 1235)

32.2.50 Math.IsZero Function

Indicates when a floating-point variable or expression evaluates to zero, or very close to zero.

Pascal

```
function IsZero(const A: Extended; Epsilon: Extended = 0): Boolean; overload;
function IsZero(const A: Double; Epsilon: Double = 0): Boolean; overload;
function IsZero(const A: Single; Epsilon: Single = 0): Boolean; overload;
```

C++

```
Boolean IsZero(const Extended A, Extended Epsilon = 0);
Boolean IsZero(const Double A, Double Epsilon = 0);
Boolean IsZero(const Single A, Single Epsilon = 0);
```

File

Math

Description

Use IsZero to test whether the value specified by AValue is zero, or differs from zero by at most Epsilon.

See Also

SameValue (see page 1251)

IsInfinite (see page 1235)

Sign (see page 1255)

32.2.51 Math.Ldexp Function

Calculates X times (2 to the power of P).

Pascal

```
function Ldexp(const X: Extended; const P: Integer): Extended;
```

C++

```
Extended Ldexp(const Extended X, const int P);
```

File

Math

Description

Ldexp returns X times (2 to the power of P).

32.2.52 Math.LnXP1 Function

Returns the natural log of (X+1)

Pascal

```
function LnXP1(const X: Extended): Extended;
```

C++

```
Extended LnXP1(const Extended X);
```

File

Math

Description

LnXP1 returns the natural logarithm of (X+1). Use LnXP1 when X is a value near 0.

32.2.53 Math.Log10 Function

Calculates log base 10.

Pascal

```
function Log10(const X: Extended): Extended;
```

C++

```
Extended Log10(const Extended X);
```

File

Math

Description

Log10 returns the log base 10 of X.

See Also

Log2 (see page 1238)

Ln

LogN (see page 1238)

32.2.54 Math.Log2 Function

Calculates log base 2.

Pascal

```
function Log2(const X: Extended): Extended;
```

C++

```
Extended Log2(const Extended X);
```

File

Math

Description

Log2 returns the log base 2 of X.

See Also

Log10 (see page 1237)

Ln

LogN (see page 1238)

32.2.55 Math.LogN Function

Calculates the log of X for a specified base.

Pascal

```
function LogN(const Base: Extended; const X: Extended): Extended;
```

C++

```
Extended LogN(const Extended Base, const Extended X);
```

File

Math

Description

LogN returns the log base Base of X.

See Also

Log2 (see page 1238)

Log10 (see page 1237)

Ln

32.2.56 Math.Max Function

Returns the greater of two numeric values (Delphi only).

Pascal

```
function Max(const A: Integer; const B: Integer): Integer; overload;
function Max(const A: Int64; const B: Int64): Int64; overload;
function Max(const A: Single; const B: Single): Single; overload;
function Max(const A: Double; const B: Double): Double; overload;
function Max(const A: Extended; const B: Extended): Extended; overload;
```

C++

```
int Max(const int A, const int B);
Int64 Max(const Int64 A, const Int64 B);
Single Max(const Single A, const Single B);
Double Max(const Double A, const Double B);
Extended Max(const Extended A, const Extended B);
```

File

Math

Description

In Delphi, call Max to compare two numeric values. Max returns the greater value of the two.

See Also

MaxValue (see page 1240)

Min (see page 1241)

MaxIntValue (see page 1239)

32.2.57 Math.MaxIntValue Function

Returns the largest signed value in an integer array.

Pascal

```
function MaxIntValue(const Data: array of Integer): Integer;
```

C++

```
int MaxIntValue(const array of Integer Data);
```

File

Math

Description

Call MaxIntValue to obtain the largest integer from the Data array. In C++, Data_Size is the index of the last element in the array (one less than the number of elements).

See Also

MaxValue (see page 1240)

MinIntValue (see page 1242)

32.2.58 Math.MaxValue Function

Returns the largest signed value in an array.

Pascal

```
function MaxValue(const Data: array of Double): Double;
```

C++

```
Double MaxValue(const array of Double Data);
```

File

Math

Description

MaxValue returns the largest signed value in the Data array. In C++, the Data_Size parameter gives the index of the last element of the array (one less than the number of elements).

See Also

MaxIntValue (see page 1239)

MinValue (see page 1242)

32.2.59 Math.Mean Function

Returns the average of all values in an array.

Pascal

```
function Mean(const Data: array of Double): Extended;
```

C++

```
Extended Mean(const array of Double Data);
```

File

Math

Description

Mean calculates the arithmetic average of all the values in the Data array parameter. In C++, the Data_Size parameter gives the index of the last element of the array (one less than the number of elements).

See Also

MeanAndStdDev (see page 1241)

32.2.60 Math.MeanAndStdDev Function

Calculates the mean and standard deviation of array elements.

Pascal

```
procedure MeanAndStdDev(const Data: array of Double; var Mean: Extended; var StdDev: Extended);
```

C++

```
MeanAndStdDev(const array of Double Data, Extended Mean, Extended StdDev);
```

File

Math

Description

MeanAndStdDev calculates Mean (see page 1240) and standard deviation in one pass. Using this is twice as fast as calculating them separately. Precision may be lost when the Mean (see page 1240) is very large (> 10e7) or the variance is very small.

Data specifies the data to be summarized. In C++, Data_Size is the index of the last element in the Data array (one less than the number of elements). The mean is returned as Mean (see page 1240) and the standard deviation is returned as StdDev (see page 1257).

Note: If the sample size is 1 (that is, if Data contains only one entry), StdDev (see page 1257) returns the value of Data[0] and not zero.

See Also

Mean (see page 1240)

StdDev (see page 1257)

32.2.61 Math.Min Function

Returns the lesser of two numeric values (Delphi only).

Pascal

```
function Min(const A: Integer; const B: Integer): Integer; overload;
function Min(const A: Int64; const B: Int64): Int64; overload;
function Min(const A: Single; const B: Single): Single; overload;
function Min(const A: Double; const B: Double): Double; overload;
function Min(const A: Extended; const B: Extended): Extended; overload;
```

C++

```
int Min(const int A, const int B);
Int64 Min(const Int64 A, const Int64 B);
Single Min(const Single A, const Single B);
Double Min(const Double A, const Double B);
Extended Min(const Extended A, const Extended B);
```

File

Math

Description

Call Min to compare two numeric values in Delphi. Min returns the smaller value of the two.

See Also

MinValue (see page 1242)

Max (see page 1239)

MinIntValue (see page 1242)

32.2.62 Math.MinIntValue Function

Returns the smallest signed value in an integer array.

Pascal

```
function MinIntValue(const Data: array of Integer): Integer;
```

C++

```
int MinIntValue(const array of Integer Data);
```

File

Math

Description

Call MinIntValue to obtain the smallest integer in the Data array. In C++, the Data_Size parameter specifies the index of the last element in the array (one less than the total number of elements).

See Also

MaxIntValue (see page 1239)

MinValue (see page 1242)

32.2.63 Math.MinValue Function

Returns smallest signed value in an array.

Pascal

```
function MinValue(const Data: array of Double): Double;
```

C++

```
Double MinValue(const array of Double Data);
```

File

Math

Description

MinValue returns the smallest signed value in the Data array. The Data_Size parameter specifies the index of the last element in the array (one less than the total number of elements).

See Also

MaxValue (see page 1240)

MinIntValue (see page 1242)

32.2.64 Math.MomentSkewKurtosis Function

Calculates the mean, variance, skew, and kurtosis.

Pascal

```
procedure MomentSkewKurtosis(const Data: array of Double; var M1: Extended; var M2: Extended;
var M3: Extended; var M4: Extended; var Skew: Extended; var Kurtosis: Extended);
```

C++

```
MomentSkewKurtosis(const array of Double Data, Extended M1, Extended M2, Extended M3, Extended
M4, Extended Skew, Extended Kurtosis);
```

File

Math

Description

MomentSkewKurtosis calculates the core factors of statistical analysis: the first four moments plus the coefficients of skew and kurtosis.

Data specifies the data to be analyzed. In C++, Data_Size is the index of the last element in the Data array (one less than the number of elements). M1, M2, M3 and M4 return the first four moments (M1 is the mean, M2 is the variance, and so on). Skew reflects symmetry of distribution. Kurtosis reflects flatness of distribution.

32.2.65 Math.NetPresentValue Function

Calculates the current value from an array of estimated cash flow values.

Pascal

```
function NetPresentValue(const Rate: Extended; const CashFlows: array of Double; PaymentTime:
TPaymentTime): Extended;
```

C++

```
Extended NetPresentValue(const Extended Rate, const array of Double CashFlows, TPaymentTime
PaymentTime);
```

File

Math

Description

NetPresentValue calculates the current value of an array of estimated cash flow values, discounted at the given interest rate of Rate.

CashFlows is the array of estimated cash flow values. In C++, CashFlows_Size is the index of the last element in that array (one less than the number of elements).

PaymentTime indicates whether the cash flows occur at the beginning or end of the period.

NetPresentValue helps determine how much an investment is currently worth, based on expected earnings, although its accuracy depends on the accuracy of the cash flows in the array.

32.2.66 Math.Norm Function

Returns the Euclidean 'L-2' norm.

Pascal

```
function Norm(const Data: array of Double): Extended;
```

C++

```
Extended Norm(const array of Double Data);
```

File

Math

Description

Norm returns the Euclidean 'L-2' norm for the values in the Data array. The 'L-2' norm is the square root of the sum of squares.

Note: In C++, Data is an array of values, and Data_Size is the index of the last value in the array (one less than the number of values).

See Also

SumOfSquares (see page 1259)

32.2.67 Math.NumberOfPeriods Function

Returns the number of payment periods for an investment.

Pascal

```
function NumberOfPeriods(const Rate: Extended; Payment: Extended; const PresentValue:
Extended; const FutureValue: Extended; PaymentTime: TPaymentTime): Extended;
```

C++

```
Extended NumberOfPeriods(const Extended Rate, Extended Payment, const Extended PresentValue,
const Extended FutureValue, TPaymentTime PaymentTime);
```

File

Math

Description

NumberOfPeriods computes the number of payment periods required for an investment of PresentValue (see page 1247) to reach a value of FutureValue (see page 1228), while making regular payments of Payment (see page 1245) and accruing interest at the rate of Rate per compounding period. PaymentTime indicates whether the cash flows occur at the beginning or end of the period.

32.2.68 Math.Payment Function

Calculates a fully amortized payment.

Pascal

```
function Payment(Rate: Extended; NPeriods: Integer; const PresentValue: Extended; const
FutureValue: Extended; PaymentTime: TPaymentTime): Extended;
```

C++

```
Extended Payment(Extended Rate, int NPeriods, const Extended PresentValue, const Extended
FutureValue, TPaymentTime PaymentTime);
```

File

Math

Description

Payment calculates the fully amortized payment of borrowing PresentValue (see page 1247) at Rate percent per period over NPeriods. It assumes that interest is paid at the end of each period.

FutureValue (see page 1228) is the value that the investment will reach at some point. PaymentTime indicates whether the cash flows occur at the beginning or end of the period.

32.2.69 Math.PeriodPayment Function

Returns the principal amount from a full payment.

Pascal

```
function PeriodPayment(const Rate: Extended; Period: Integer; NPeriods: Integer; const
PresentValue: Extended; const FutureValue: Extended; PaymentTime: TPaymentTime): Extended;
```

C++

```
Extended PeriodPayment(const Extended Rate, int Period, int NPeriods, const Extended
PresentValue, const Extended FutureValue, TPaymentTime PaymentTime);
```

File

Math

Description

PeriodPayment gives the part of the payment that is principal. The InterestPayment (see page 1233) function gives the part of the payment that is interest.

The Rate parameter is the interest rate. Period is the number of periods into the loan for which the principal is desired, and NPeriods is the number of periods of the loan. FutureValue (see page 1228) is the value the investment will reach at some point. PaymentTime indicates whether the cash flows occur at the beginning or end of the period.

32.2.70 Math.Poly Function

Evaluates a uniform polynomial of one variable at the value X.

Pascal

```
function Poly(const X: Extended; const Coefficients: array of Double): Extended;
```

C++

```
Extended Poly(const Extended X, const array of Double Coefficients);
```

File

Math

Description

Call Poly to evaluate the polynomial represented by the Coefficients parameter at the point where the variable equals the value of the X parameter. The coefficients are ordered in increasing powers of X, as shown in the following Delphi-like example:

Coefficients[0] + Coefficients[1]*X + ... + Coefficients[N]*(X**N)

Note: In C++, the Coefficients_Size parameter is N, the highest power to which X is raised.

32.2.71 Math.PopnStdDev Function

Calculates the population standard deviation.

Pascal

```
function PopnStdDev(const Data: array of Double): Extended;
```

C++

```
Extended PopnStdDev(const array of Double Data);
```

File

Math

Description

PopnStdDev calculates the population standard deviation (the square root of the population variance) of all values in the Data array parameter. In C++, the Data_Size parameter indicates the index of the last element in Data (one less than the number of data points).

This function is used in some business and financial calculations.

See Also

StdDev (see page 1257)

PopnVariance (see page 1246)

32.2.72 Math.PopnVariance Function

Calculates the population variance.

Pascal

```
function PopnVariance(const Data: array of Double): Extended;
```

C++

```
Extended PopnVariance(const array of Double Data);
```

File

Math

Description

PopnVariance calculates the population variance of all values in the Data array parameter, using the n method (biased): TotalVariance (see page 1261) / n. In C++,. the Data_Size parameter is the index of the last value in the Data array (one less than the number of data items).

See Also

Variance (see page 1262)

TotalVariance (see page 1261)

PopnStdDev (see page 1246)

32.2.73 Math.Power Function

Raises Base to any power.

Pascal

```
function Power(const Base: Extended; const Exponent: Extended): Extended; overload;
function Power(const Base: Double; const Exponent: Double): Double; overload;
function Power(const Base: Single; const Exponent: Single): Single; overload;
```

C++

```
Extended Power(const Extended Base, const Extended Exponent);
Double Power(const Double Base, const Double Exponent);
Single Power(const Single Base, const Single Exponent);
```

File

Math

Description

Power raises Base to any power. For fractional exponents or exponents greater than MaxInt, Base must be greater than 0.

See Also

IntPower (see page 1235)

Sqrt

32.2.74 Math.PresentValue Function

Calculates the present value of an investment.

Pascal

```
function PresentValue(const Rate: Extended; NPeriods: Integer; const Payment: Extended; const
FutureValue: Extended; PaymentTime: TPaymentTime): Extended;
```

C++

```
Extended PresentValue(const Extended Rate, int NPeriods, const Extended Payment, const
Extended FutureValue, TPaymentTime PaymentTime);
```

File

Math

Description

PresentValue calculates the present value of an investment where Payment (see page 1245) is received for NPeriods and is discounted at the rate of Rate per period. FutureValue (see page 1228) is the value the investment may reach at some point. PaymentTime indicates whether the cash flows occur at the beginning or end of the period.

32.2.75 Math.RadToCycle Function

Converts radians to cycles.

Pascal

```
function RadToCycle(const Radians: Extended): Extended;
```

C++

```
Extended RadToCycle(const Extended Radians);
```

File

Math

Description

Use RadToCycle to convert angles measured in radians into cycles, where

cycles = radians/(2pi).

32.2.76 Math.RadToDeg Function

Converts radians to degrees.

Pascal

```
function RadToDeg(const Radians: Extended): Extended;
```

C++

```
Extended RadToDeg(const Extended Radians);
```

File

Math

Description

Use RadToDeg to convert angles measured in radians to degrees, where

degrees = radians(180/pi).

32.2.77 Math.RadToGrad Function

Converts radians to grads.

Pascal

```
function RadToGrad(const Radians: Extended): Extended;
```

C++

```
Extended RadToGrad(const Extended Radians);
```

File

Math

Description

Use RadToGrad to convert angles measured in radians to grads, where

grads = radians(200/pi).

32.2.78 Math.RandG Function

Generates random numbers with Gaussian distribution.

Pascal

```
function RandG(Mean: Extended; StdDev: Extended): Extended;
```

C++

```
Extended RandG(Extended Mean, Extended StdDev);
```

File

Math

Description

RandG produces random numbers with Gaussian distribution about the Mean (see page 1240). This is useful for simulating data with sampling errors and expected deviations from the Mean (see page 1240).

32.2.79 Math.RandomFrom Function

Returns a randomly selected element from an array.

Pascal

```
function RandomFrom(const AValues: array of Integer): Integer; overload;
function RandomFrom(const AValues: array of Int64): Int64; overload;
function RandomFrom(const AValues: array of Double): Double; overload;
```

C++

```
int RandomFrom(const array of Integer AValues);
Int64 RandomFrom(const array of Int64 AValues);
Double RandomFrom(const array of Double AValues);
```

File

Math

Description

RandomFrom returns a randomly selected element from the array passed as the AValues argument. In C++, AValues_Size is

the index of the last array element (one less than the number of elements.)

To initialize the random number generator, add a single call Randomize or assign a value to the RandSeed variable before making any calls to RandomFrom.

See Also

Randomize

RandSeed

32.2.80 Math.RandomRange Function

Returns a random integer from a specified range.

Pascal

```
function RandomRange(const AFrom: Integer; const ATo: Integer): Integer;
```

C++

```
int RandomRange(const int AFrom, const int ATo);
```

File

Math

Description

RandomRange returns a random integer from the range that extends between AFrom and ATo (non-inclusive). RandomRange can handle negative ranges (where AFrom is greater than ATo).

To initialize the random number generator, add a single call Randomize or assign a value to the RandSeed variable before making any calls to RandomRange.

See Also

Randomize

RandSeed

RandomFrom (see page 1249)

Random

32.2.81 Math.RoundTo Function

Rounds a floating-point value to a specified digit or power of ten using "Banker's rounding".

Pascal

```
function RoundTo(const AValue: Extended; const ADigit: TRoundToEXRangeExtended): Extended;
```

C++

```
Extended RoundTo(const Extended AValue, const TRoundToEXRangeExtended ADigit);
```

File

Math

Description

Call RoundTo to round AValue to a specified power of ten.

AValue is the value to round.

ADigit indicates the power of ten to which you want AValue rounded. It can be any value from –37 to 37 (inclusive).

RoundTo uses "Banker's Rounding" to determine how to round values that are exactly midway between the two values that have the desired number of significant digits. This method rounds to an even number in the case that AValue is not nearer to either value.

The following examples illustrate the use of RoundTo:

Expression	Value
RoundTo(1234567, 3)	1235000
RoundTo(1.234, -2)	1.23
RoundTo(1.235, -2)	1.24
RoundTo(1.245, -2)	1.24

Note: The behavior of RoundTo can be affected by the Set8087CW procedure or SetRoundMode (see page 1254) function.

See Also

Round

SetRoundMode (see page 1254)

Set8087CW

SimpleRoundTo (see page 1255)

32.2.82 Math.SameValue Function

Indicates whether two floating-point values are (approximately) equal.

Pascal

```
function SameValue(const A: Extended; const B: Extended; Epsilon: Extended = 0): Boolean;
overload;
function SameValue(const A: Double; const B: Double; Epsilon: Double = 0): Boolean; overload;
function SameValue(const A: Single; const B: Single; Epsilon: Single = 0): Boolean; overload;
```

C++

```
Boolean SameValue(const Extended A, const Extended B, Extended Epsilon = 0);
Boolean SameValue(const Double A, const Double B, Double Epsilon = 0);
Boolean SameValue(const Single A, const Single B, Single Epsilon = 0);
```

File

Math

Description

Call SameValue to determine when two floating-point values are equal, or approximately equal.

A and B are the values to compare.

Epsilon is the maximum amount by which A and B can differ and still be considered the same value.

See Also

CompareValue (see page 1220)

IsZero (see page 1236)

32.2.83 Math.Sec Function

Calculates the secant of an angle.

Pascal

```
function Sec(const X: Extended): Extended;
```

C++

```
Extended Sec(const Extended X);
```

File

Math

Description

Call Sec to obtain the secant of X, where X is an angle in radians. The secant is calculated using the formula 1 / Cos(X).

Note: Sec is identical to the Secant (see page 1252) function.

See Also

ArcSec (see page 1216)

Secant (see page 1252)

SecH (see page 1253)

Cos

32.2.84 Math.Secant Function

Calculates the secant of an angle.

Pascal

```
function Secant(const X: Extended): Extended;
```

C++

```
Extended Secant(const Extended X);
```

File

Math

Description

Call Secant to obtain the secant of X, where X is an angle in radians. The secant is calculated using the formula 1 / Cos(X).

See Also

Sec (see page 1252)

SecH (see page 1253)

Cos

32.2.85 Math.SecH Function

Calculates the hyperbolic secant of an angle.

Pascal

```
function SecH(const X: Extended): Extended;
```

C++

```
Extended SecH(const Extended X);
```

File

Math

Description

Call SecH to obtain the hyperbolic secant of X, where X is an angle in Radians.

See Also

Sec (see page 1252)

ArcSec (see page 1216)

32.2.86 Math.SetExceptionMask Function

Sets the exception mask on the FPU control word.

Pascal

```
function SetExceptionMask(const Mask: TFPUExceptionMask): TFPUExceptionMask;
```

C++

```
TFPUExceptionMask SetExceptionMask(const TFPUExceptionMask Mask);
```

File

Math

Description

Call SetExceptionMask to specify the types of exceptions the FPU (floating-point unit) generates. Any element set in this mask prevents the FPU from raising the corresponding type of exception. Instead, the FPU returns its best attempt at a value – often NaN (see page 1274), Infinity (see page 1270), or NegInfinity (see page 1275). The value depends on the operation and the current rounding mode.

SetExceptionMask returns the previous exception mask.

See Also

Set8087CW

GetExceptionMask (see page 1229)

NaN (see page 1274)

Infinity (see page 1270)

NegInfinity (see page 1275)

32.2.87 Math.SetPrecisionMode Function

Sets the FPU precision control mode.

Pascal

```
function SetPrecisionMode(const Precision: TFPUPrecisionMode): TFPUPrecisionMode;
```

C++

```
TFPUPrecisionMode SetPrecisionMode(const TFPUPrecisionMode Precision);
```

File

Math

Description

Call SetPrecisionMode to specify the level of precision that the FPU (floating-point unit) uses for floating-point calculations. The precision control mode can be any of the following values:

Value	Meaning
pmSingle	single precision
pmReserved	not used
pmDouble	double precision
pmExtended	extended precision

SetPrecisionMode returns the previous precision control mode.

See Also

Set8087CW

GetPrecisionMode (see page 1229)

32.2.88 Math.SetRoundMode Function

Sets the FPU rounding mode.

Pascal

```
function SetRoundMode(const RoundMode: TFPURoundingMode): TFPURoundingMode;
```

C++

```
TFPURoundingMode SetRoundMode(const TFPURoundingMode RoundMode);
```

File

Math

Description

Call SetRoundingMode to specify how the FPU handles rounding issues. The rounding mode can be any of the following values:

Value	Meaning
rmNearest	Rounds to the closest value.
rmDown	Rounds toward negative infinity.
rmUp	Rounds toward positive infinity.
rmTruncate	Truncates the value, rounding positive numbers down and negative numbers up.

See Also

Set8087CW

GetRoundMode (see page 1230)

32.2.89 Math.Sign Function

Indicates whether a numeric value is positive, negative, or zero.

Pascal

```
function Sign(const AValue: Integer): TValueSign; overload;
function Sign(const AValue: Int64): TValueSign; overload;
function Sign(const AValue: Double): TValueSign; overload;
```

C++

```
TValueSign Sign(const int AValue);
TValueSign Sign(const Int64 AValue);
TValueSign Sign(const Double AValue);
```

File

Math

Description

Use Sign to test the sign of a numeric value. Sign returns

0 if AValue is zero.

1 if AValue is greater than zero.

-1 if AValue is less than zero.

See Also

IsZero (see page 1236)

32.2.90 Math.SimpleRoundTo Function

Rounds a floating-point value to a specified digit or power of ten using asymmetric arithmetic rounding.

Pascal

```
function SimpleRoundTo(const AValue: Extended; const ADigit: TRoundToRange = -2): Extended;
```

C++

```
Extended SimpleRoundTo(const Extended AValue, const TRoundToRange ADigit = -2);
```

File

Math

Description

Call SimpleRoundTo to round AValue to a specified power of ten.

AValue is the value to round.

ADigit indicates the power of ten to which you want AValue rounded. It can be any value from –37 to 37 (inclusive).

SimpleRoundTo uses asymmetric arithmetic rounding to determine how to round values that are exactly midway between the two values that have the desired number of significant digits. This method always rounds to the larger value.

The following examples illustrate the use of SimpleRoundTo:

Expression	Value
SimpleRoundTo(1234567, 3)	1234000
SimpleRoundTo(1.234, -2)	1.23
SimpleRoundTo(1.235, -2)	1.24
SimpleRoundTo(-1.235, -2)	-1.23

See Also

Round

RoundTo (see page 1250)

32.2.91 Math.SinCos Function

Returns sine and cosine of an angle.

Pascal

```
procedure SinCos(const Theta: Extended; var Sin: Extended; var Cos: Extended);
```

C++

```
SinCos(const Extended Theta, Extended Sin, Extended Cos);
```

File

Math

Description

Call SinCos to obtain the sine and cosine of an angle expressed in radians. Theta specifies the angle. The sine and cosine are returned in the Sin and Cos parameters, respectively.

SinCos is twice as fast as calling Sin and Cos separately for the same angle.

32.2.92 Math.Sinh Function

Returns the hyperbolic sine of an angle.

Pascal

```
function Sinh(const X: Extended): Extended;
```

C++

```
Extended Sinh(const Extended X);
```

File

Math

Description

Sinh calculates the hyperbolic sine of X.

32.2.93 Math.SLNDepreciation Function

Returns the straight-line depreciation allowance of an asset.

Pascal

```
function SLNDepreciation(const Cost: Extended; const Salvage: Extended; Life: Integer):
Extended;
```

C++

```
Extended SLNDepreciation(const Extended Cost, const Extended Salvage, int Life);
```

File

Math

Description

SLNDepreciation calculates the straight-line depreciation allowance for an asset over one period of its life. The function divides the Cost minus the Salvage by the number of years of useful Life of the asset. Cost is the amount initially paid for the asset. Salvage is the value of the asset at the end of its useful life.

To compute accelerated depreciation (allowing higher depreciation values in the first years of the assets life), use the SYDDepreciation (see page 1260) function.

See Also

SYDDepreciation (see page 1260)

32.2.94 Math.StdDev Function

Returns the sample standard deviation for elements in an array.

Pascal

```
function StdDev(const Data: array of Double): Extended;
```

C++

```
Extended StdDev(const array of Double Data);
```

File

Math

Description

StdDev calculates the sample standard deviation (the square root of the sample variance) of all values in the Data array parameter. In C++, Data_Size indicates the index of the last element in the array (one less than the number of elements).

See Also

TotalVariance (see page 1261)

Variance (see page 1262)

32.2.95 Math.Sum Function

Returns the sum of the elements in an array.

Pascal

```
function Sum(const Data: array of Double): Extended;
```

C++

```
Extended Sum(const array of Double Data);
```

File

Math

Description

Sum returns the sum of all the values in the Data array parameter. In C++, the Data_Size parameter specifies the index of the last value in Data (one less than the number of values).

See Also

SumInt (see page 1258)

32.2.96 Math.SumInt Function

Returns the sum of the elements in an integer array.

Pascal

```
function SumInt(const Data: array of Integer): Integer;
```

C++

```
int SumInt(const array of Integer Data);
```

File

Math

Description

Sum (see page 1258) returns the sum of all the values in the Data array parameter. In C++, the Data_Size parameter specifies the index of the last value in Data (one less than the number of values).

See Also

Sum (see page 1258)

32.2.97 Math.SumOfSquares Function

Returns the sum of the squared values from a data array.

Pascal

```
function SumOfSquares(const Data: array of Double): Extended;
```

C++

```
Extended SumOfSquares(const array of Double Data);
```

File

Math

Description

SumOfSquares returns the sum of the squares of the Data values: Data[0]^2 + Data[1]^2 + Data[2]^2... In C++, the Data_Size parameter specifies the index of the last value in Data (one less than the number of values).

See Also

Sum (see page 1258)

SumsAndSquares (see page 1259)

32.2.98 Math.SumsAndSquares Function

Returns the sum of the values and the sum of the squared values in an array.

Pascal

```
procedure SumsAndSquares(const Data: array of Double; var Sum: Extended; var SumOfSquares:
Extended);
```

C++

```
SumsAndSquares(const array of Double Data, Extended Sum, Extended SumOfSquares);
```

File

Math

Description

SumsAndSquares returns the sum of the Data values in Sum (see page 1258) and the sum of the squares in SumOfSquares (see page 1259). The data to sum is specified in the Data array parameter. In C++, Data_Size indicates the index of the last value (one less than the number of values in Data).

This function is faster than calculating the sums separately.

See Also

Sum (see page 1258)

SumOfSquares (see page 1259)

TotalVariance (see page 1261)

32.2.99 Math.SYDDepreciation Function

Calculates depreciation for an asset.

Pascal

```
function SYDDepreciation(const Cost: Extended; const Salvage: Extended; Life: Integer; Period: Integer): Extended;
```

C++

```
Extended SYDDepreciation(const Extended Cost, const Extended Salvage, int Life, int Period);
```

File

Math

Description

SYDDepreciation (for "sum-of-years-digits depreciation") calculates depreciation amounts for an asset using an accelerated depreciation method. This allows for higher depreciation in the earlier years of an asset's life.

Cost is the initial cost of the asset. Salvage is the value of the asset at the end of its life expectancy. Life is the length of the asset's life expectancy. Period is the period for which to calculate the depreciation.

See Also

SLNDepreciation (see page 1257)

32.2.100 Math.Tan Function

Returns the tangent of X.

Pascal

```
function Tan(const X: Extended): Extended;
```

C++

```
Extended Tan(const Extended X);
```

File

Math

Description

Tan returns the tangent of X. Tan(X) = Sin(X) / Cos(X).

See Also

Sin

Cos

ArcTan

32.2.101 Math.Tanh Function

Returns the hyperbolic tangent of X.

Pascal

```
function Tanh(const X: Extended): Extended;
```

C++

```
Extended Tanh(const Extended X);
```

File

Math

Description

Tanh calculates the hyperbolic tangent of X.

See Also

Cosh (see page 1221)

Sinh (see page 1256)

32.2.102 Math.TotalVariance Function

Returns the statistical variance from an array of values.

Pascal

```
function TotalVariance(const Data: array of Double): Extended;
```

C++

```
Extended TotalVariance(const array of Double Data);
```

File

Math

Description

TotalVariance returns the sum of the squares of the difference between each Data value and the mean of the data values.

Total Variance (see page 1262): SUM(i=1,N)[(X(i) - Mean (see page 1240))**2]

See Also

SumOfSquares (see page 1259)

Variance (see page 1262)

StdDev (see page 1257)

Mean (see page 1240)

32.2.103 Math.Variance Function

Calculates statistical sample variance from an array of data.

Pascal

```
function Variance(const Data: array of Double): Extended;
```

C++

```
Extended Variance(const array of Double Data);
```

File

Math

Description

Variance calculates the sample variance of all values in the Data array parameter, using the n-1 method (unbiased). In C++, Data_Size specifies the index of the last value in Data (on less than the number of data values).

See Also

TotalVariance (see page 1261)

StdDev (see page 1257)

Mean (see page 1240)

PopnVariance (see page 1246)

32.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TFPUException (see page 1263)	TFPUException defines TFPUExceptionMask (see page 1267) values.
	TFPUPrecisionMode (see page 1264)	TFPUPrecisionMode defines how the FPU handles precision.
	TFPURoundingMode (see page 1265)	TFPURoundingMode defines how the FPU handles rounding issues.
	TPaymentTime (see page 1266)	TPaymentTime is an enumerated type used as a parameter in financial functions.

32.3.1 Math.TFPUException Enumeration

TFPUException defines TFPUExceptionMask (see page 1267) values.

Pascal

```
TFPUException = (
  exInvalidOp,
  exDenormalized,
  exZeroDivide,
  exOverflow,
  exUnderflow,
  exPrecision
);
```

C++

```
enum TFPUException {
  exInvalidOp,
  exDenormalized,
  exZeroDivide,
  exOverflow,
  exUnderflow,
  exPrecision
};
```

File

Math

Description

The FPU (Floating Point Unit) handles floating point calculations. You can prevent exceptions from occuring by using SetExceptionMask (see page 1253) to mask some or all of them.

When an exception is masked, the FPU returns its best attempt at a value – often NaN (see page 1274), Infinity (see page 1270), or NegInfinity (see page 1275). The value depends on the operation and the current rounding mode (see GetRoundMode (see page 1230) and SetRoundMode (see page 1254)).

TFPUExceptionMask (see page 1267) can have none, one, some or all of the following TFPUException values set :

Value	Meaning
exInvalidOp	An invalid operation was attempted.
exDenormalized	A number was reduced in size smaller than can be stored as non-zero. It has been denormalized.

exZeroDivide	An attempt was made to divide by zero.
exOverflow	A number has exceeded the highest positive value supported.
exUnderflow	A number has exceeded the highest negative value supported.
exPrecision	A number has exceeded the number of digits of precision.

See Also

GetExceptionMask (see page 1229)

GetRoundMode (see page 1230)

SetExceptionMask (see page 1253)

SetRoundMode (see page 1254)

TFPUExceptionMask (see page 1267)

32.3.2 Math.TFPUPrecisionMode Enumeration

TFPUPrecisionMode defines how the FPU handles precision.

Pascal

```
TFPUPrecisionMode = (
  pmSingle,
  pmReserved,
  pmDouble,
  pmExtended
);
```

C++

```
enum TFPUPrecisionMode {
  pmSingle,
  pmReserved,
  pmDouble,
  pmExtended
};
```

File

Math

Description

The FPU (Floating Point Unit) handles floating point calculations. GetPrecisionMode (see page 1229) is used to read, and SetPrecisionMode (see page 1254) is used to set the precision control mode, with TFPUPrecisionMode defining this mode.

TFPUPrecisionMode can be one of the following values :

Value	Meaning
pmSingle	Single precision : 7 significant digits, exponent from -38 to +38.
pmReserved	Reserved for future use.
pmDouble	Double precision : 15 significant digits, exponent from -308 to +308.
pmExtended	Extended precision : 19 significant digits, exponent from -4932 to +4932.

See Also

GetPrecisionMode (see page 1229)

GetRoundMode (see page 1230)

SetPrecisionMode (see page 1254)

SetRoundMode (see page 1254)

32.3.3 Math.TFPURoundingMode Enumeration

TFPURoundingMode defines how the FPU handles rounding issues.

Pascal

```
TFPURoundingMode = (
  rmNearest,
  rmDown,
  rmUp,
  rmTruncate
);
```

C++

```
enum TFPURoundingMode {
  rmNearest,
  rmDown,
  rmUp,
  rmTruncate
};
```

File

Math

Description

The FPU (Floating Point Unit) handles floating point calculations. GetRoundMode (see page 1230) is used to read, and SetRoundMode (see page 1254) is used to set the rounding mode, with TFPURoundingMode defining this mode.

TFPURoundingMode can be one of the following values :

Value	Meaning
rmNearest	Rounds to the closest value.
rmDown	Rounds toward negative infinity.
rmUp	Rounds toward positive infinity.
rmTruncate	Truncates the value, rounding positive numbers down and negative numbers up.

See Also

GetPrecisionMode (see page 1229)

GetRoundMode (see page 1230)

SetPrecisionMode (see page 1254)

SetRoundMode (see page 1254)

32.3.4 Math.TPaymentTime Enumeration

TPaymentTime is an enumerated type used as a parameter in financial functions.

Pascal

```
TPaymentTime = (
  ptEndOfPeriod,
  ptStartOfPeriod
);
```

C++

```
enum TPaymentTime {
  ptEndOfPeriod,
  ptStartOfPeriod
};
```

File

Math

Description

TPaymentTime indicates when interest is calculated. The following table lists the possible values:

Value	Meaning
ptStartOfPeriod	Interest is calculated on the principle balance at the beginning of the payment period
ptEndOfPeriod	Interest is calculated at the end of the payment period

32.4 Types

The following table lists types in this documentation.

Types

Name	Description
TFPUExceptionMask (see page 1267)	TFPUExceptionMask defines a set of TFPUException (see page 1263) values.
TRoundToEXRangeExtended (see page 1268)	This is type Math.TRoundToEXRangeExtended.
TRoundToRange (see page 1268)	TRoundToRange defines the RoundTo (see page 1250) rounding limits.
TValueSign (see page 1268)	TValueSign defines the Sign (see page 1255) returned value.

32.4.1 Math.TFPUExceptionMask Type

TFPUExceptionMask defines a set of TFPUException (see page 1263) values.

Pascal

```
TFPUExceptionMask = set of TFPUException;
```

C++

```
TFPUException TFPUExceptionMask;
```

File

Math

Description

The FPU (Floating Point Unit) handles floating point calculations. You can prevent exceptions from occuring by using SetExceptionMask (see page 1253) to mask some or all of them.

When an exception is masked, the FPU returns its best attempt at a value – often NaN (see page 1274), Infinity (see page 1270), or NegInfinity (see page 1275). The value depends on the operation and the current rounding mode (see GetRoundMode (see page 1230) and SetRoundMode (see page 1254)).

TFPUExceptionMask can have none, one, some or all of the following TFPUException (see page 1263) values set :

Value	Meaning
exInvalidOp	An invalid operation was attempted.
exDenormalized	A number was reduced in size smaller than can be stored as non-zero. It has been denormalized.
exZeroDivide	An attempt was made to divide by zero.
exOverflow	A number has exceeded the highest positive value supported.
exUnderflow	A number has exceeded the highest negative value supported.
exPrecision	A number has exceeded the number of digits of precision.

See Also

GetExceptionMask (see page 1229)

GetRoundMode (see page 1230)

SetExceptionMask (see page 1253)

SetRoundMode (see page 1254)

TFPUException (see page 1263)

32.4.2 Math.TRoundToEXRangeExtended Type

Pascal

```
TRoundToEXRangeExtended = -20..20;
```

C++

```
-20..20 TRoundToEXRangeExtended;
```

File

Math

Description

This is type Math.TRoundToEXRangeExtended.

32.4.3 Math.TRoundToRange Type

TRoundToRange defines the RoundTo (see page 1250) rounding limits.

Pascal

```
TRoundToRange = -37..37;
```

C++

```
-37..37 TRoundToRange;
```

File

Math

Description

The RoundTo (see page 1250) routine rounds a floating-point value to a specified digit or power of ten using "Banker's rounding". A negative TRoundToRange value rounds to this number of decimal places. A positive value removes all decimal places, and zeroes this number of significant digits to the left of the decimal point. Banker's rounding always rounds down to an even value last digit when it has a choice.

See Also

RoundTo (see page 1250)

32.4.4 Math.TValueSign Type

TValueSign defines the Sign (see page 1255) returned value.

Pascal

```
TValueSign = -1..1;
```

C++

```
-1..1 TValueSign;
```

File

Math

Description

The Sign (see page 1255) routine determines whether a number is negative, zero or positive. It returns the result in a TValueSign value as follows :

Value	Associated constant	Meaning
-1	NegativeValue (see page 1274)	Negative number.
0	ZeroValue (see page 1276)	Zero.
+1	PositiveValue (see page 1275)	Positive number.

See Also

NegativeValue (see page 1274)

PositiveValue (see page 1275)

ZeroValue (see page 1276)

RoundTo (see page 1250)

32.5 Constants

The following table lists constants in this documentation.

Constants

Name	Description
Infinity (see page 1270)	Infinity represents positive infinity.
MaxComp (see page 1271)	MaxComp defines the maximum absolute value for a Comp variable.
MaxDouble (see page 1271)	MaxDouble defines the maximum absolute value for a Double variable.
MaxExtended (see page 1271)	MaxExtended defines the maximum absolute value for an Extended variable.
MaxSingle (see page 1272)	MaxSingle defines the maximum absolute value for a Single variable.
MinComp (see page 1272)	MinComp defines the smallest absolute value for a Comp variable.
MinDouble (see page 1273)	MinDouble defines the smallest absolute value for a Double variable.
MinExtended (see page 1273)	MinExtended defines the smallest absolute value for an Extended variable.
MinSingle (see page 1273)	MinSingle defines the smallest absolute value for a Single variable.
NaN (see page 1274)	NaN represents a value that cannot be represented as a valid number.
NegativeValue (see page 1274)	NegativeValue defines a negative number result from using Sign (see page 1255).
NegInfinity (see page 1275)	NegInfinity represents negative infinity.
PositiveValue (see page 1275)	PositiveValue defines a positive number result from using Sign (see page 1255).
ZeroValue (see page 1276)	ZeroValue defines a zero value number result from using Sign (see page 1255).

32.5.1 Math.Infinity Constant

Infinity represents positive infinity.

Pascal

```
Infinity = 1.0 / 0.0;
```

C++

```
Infinity = 1.0 / 0.0;
```

File

Math

Description

Infinity is literally defined as 1.0/0.0 : positive infinity.

Note: Only use Infinity in assignments : use IsInfinite (see page 1235).

See Also

IsInfinite (see page 1235)

NegInfinity (see page 1275)

32.5.2 Math.MaxComp Constant

MaxComp defines the maximum absolute value for a Comp variable.

Pascal

```
MaxComp = 9.223372036854775807e+18;
```

C++

```
MaxComp = 9.223372036854775807e+18;
```

File

Math

Description

MaxComp defines the maximum absolute value for a Comp variable.

See Also

MinComp (see page 1272)

32.5.3 Math.MaxDouble Constant

MaxDouble defines the maximum absolute value for a Double variable.

Pascal

```
MaxDouble = 1.7e+308;
```

C++

```
MaxDouble = 1.7e+308;
```

File

Math

Description

MaxDouble defines the maximum absolute value for a Double variable.

See Also

MinDouble (see page 1273)

32.5.4 Math.MaxExtended Constant

MaxExtended defines the maximum absolute value for an Extended variable.

Pascal

```
MaxExtended = 1.1e+4932;
```

C++

```
MaxExtended = 1.1e+4932;
```

File

Math

Description

MaxExtended defines the maximum absolute value for an Extended variable.

See Also

MinExtended (see page 1273)

32.5.5 Math.MaxSingle Constant

MaxSingle defines the maximum absolute value for a Single variable.

Pascal

```
MaxSingle = 3.4e+38;
```

C++

```
MaxSingle = 3.4e+38;
```

File

Math

Description

MaxSingle defines the maximum absolute value for a Single variable.

See Also

MinSingle (see page 1273)

32.5.6 Math.MinComp Constant

MinComp defines the smallest absolute value for a Comp variable.

Pascal

```
MinComp = -9.223372036854775807e+18;
```

C++

```
MinComp = -9.223372036854775807e+18;
```

File

Math

Description

MinComp defines the smallest absolute value for a Comp variable.

See Also

MaxComp (see page 1271)

32.5.7 Math.MinDouble Constant

MinDouble defines the smallest absolute value for a Double variable.

Pascal

```
MinDouble = 5.0e-324;
```

C++

```
MinDouble = 5.0e-324;
```

File

Math

Description

MinDouble defines the smallest absolute value for a Double variable.

See Also

MaxDouble (see page 1271)

32.5.8 Math.MinExtended Constant

MinExtended defines the smallest absolute value for an Extended variable.

Pascal

```
MinExtended = 3.4e-4932;
```

C++

```
MinExtended = 3.4e-4932;
```

File

Math

Description

MinExtended defines the smallest absolute value for an Extended variable.

See Also

MaxExtended (see page 1271)

32.5.9 Math.MinSingle Constant

MinSingle defines the smallest absolute value for a Single variable.

Pascal

```
MinSingle = 1.5e-45;
```

C++

```
MinSingle = 1.5e-45;
```

File

Math

Description

MinSingle defines the smallest absolute value for a Single variable.

See Also

MaxSingle (see page 1272)

32.5.10 Math.NaN Constant

NaN represents a value that cannot be represented as a valid number.

Pascal

```
NaN = 0.0 / 0.0;
```

C++

```
NaN = 0.0 / 0.0;
```

File

Math

Description

NaN is literally defined as 0.0/0.0 : an undefined value.

Note: Only use NaN in assignments : use IsNan (see page 1236).

See Also

IsNan (see page 1236)

32.5.11 Math.NegativeValue Constant

NegativeValue defines a negative number result from using Sign (see page 1255).

Pascal

```
NegativeValue = Low(TValueSign);
```

C++

```
NegativeValue = Low(TValueSign);
```

File

Math

Description

Sign (see page 1255) determines the sign of a number, returning the result in a TValueSign (see page 1268) value. TValueSign (see page 1268) will be NegativeValue if the number is negative.

See Also

PositiveValue (see page 1275)

Sign (see page 1255)

TValueSign (see page 1268)

ZeroValue (see page 1276)

32.5.12 Math.NegInfinity Constant

NegInfinity represents negative infinity.

Pascal

```
NegInfinity = -1.0 / 0.0;
```

C++

```
NegInfinity = -1.0 / 0.0;
```

File

Math

Description

NegInfinity is literally defined as -1.0/0.0 : negative infinity.

Note: Only use NegInfinity in assignments : use IsInfinite (see page 1235).

See Also

Infinity (see page 1270)

IsInfinite (see page 1235)

32.5.13 Math.PositiveValue Constant

PositiveValue defines a positive number result from using Sign (see page 1255).

Pascal

```
PositiveValue = High(TValueSign);
```

C++

```
PositiveValue = High(TValueSign);
```

File

Math

Description

Sign (see page 1255) determines the sign of a number, returning the result in a TValueSign (see page 1268) value. TValueSign (see page 1268) will be PositiveValue if the number is positive.

See Also

NegativeValue (see page 1274)

Sign (see page 1255)

TValueSign (see page 1268)

ZeroValue (see page 1276)

32.5.14 Math.ZeroValue Constant

ZeroValue defines a zero value number result from using Sign (see page 1255).

Pascal

```
ZeroValue = 0;
```

C++

```
ZeroValue = 0;
```

File

Math

Description

Sign (see page 1255) determines the sign of a number, returning the result in a TValueSign (see page 1268) value. TValueSign (see page 1268) will be ZeroValue if the number is zero.

See Also

NegativeValue (see page 1274)

PositiveValue (see page 1275)

Sign (see page 1255)

TValueSign (see page 1268)

33 MConnect Namespace

This is namespace MConnect.

Topics

Name	Description
Classes (see page 1278)	The following table lists classes in this documentation.
Functions (see page 1317)	The following table lists functions in this documentation.

Classes

	Name	Description
	TCOMConnection (see page 1278)	TCOMConnection is the base class for components that connect to application servers using DCOM or OLE.
	TCustomObjectBroker (see page 1281)	TCustomObjectBroker is the base class for components that locate available application servers for a DataSnap connection component.
	TDCOMConnection (see page 1284)	TDCOMConnection handles a client application's DCOM connection to the remote server in a in a multi-tiered application.
	TDispatchAppServer (see page 1289)	This is class MConnect.TDispatchAppServer.
	TDispatchConnection (see page 1290)	TDispatchConnection is the base class for objects that connect client applications to an IDispatch-based application server.
	TOLEnterpriseConnection (see page 1305)	TOLEnterpriseConnection is provided for backward compatibility with applications that connect to an application server using OLEnterprise.
	TSharedConnection (see page 1309)	TSharedConnection connects to a child remote data module that is accessible only through the properties of a main remote data module on the application server.

Functions

	Name	Description
	GetMIDASAppServerList (see page 1317)	This is function MConnect.GetMIDASAppServerList.

33.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TCOMConnection (see page 1278)	TCOMConnection is the base class for components that connect to application servers using DCOM or OLE.
	TCustomObjectBroker (see page 1281)	TCustomObjectBroker is the base class for components that locate available application servers for a DataSnap connection component.
	TDCOMConnection (see page 1284)	TDCOMConnection handles a client application's DCOM connection to the remote server in a in a multi-tiered application.
	TDispatchAppServer (see page 1289)	This is class MConnect.TDispatchAppServer.
	TDispatchConnection (see page 1290)	TDispatchConnection is the base class for objects that connect client applications to an IDispatch-based application server.
	TOLEnterpriseConnection (see page 1305)	TOLEnterpriseConnection is provided for backward compatibility with applications that connect to an application server using OLEnterprise.
	TSharedConnection (see page 1309)	TSharedConnection connects to a child remote data module that is accessible only through the properties of a main remote data module on the application server.

33.1.1 TCOMConnection Class

TCOMConnection is the base class for components that connect to application servers using DCOM or OLE.

Class Hierarchy

Pascal

```
TCOMConnection = class(TDispatchConnection);
```

C++

```
class TCOMConnection : public TDispatchConnection;
```

File

MConnect

Description

Do not create instances of TCOMConnection. Instead, use TCOMConnection as a base class when created a custom component that forms a COM-based connection to an application server.

TCOMConnection adds to the ability to work with COM objects to the generic connection capabilities inherited from TDispatchConnection (see page 1290). It has two descendants: TDCOMConnection (see page 1284), which creates and manages connections using DCOM, and TOLEnterpriseConnection (see page 1305), which is provided for backward compatibility and handles connections using OLEnterprise.

See Also

TClientDataSet

TDCOMConnection (see page 1284)

TOLEnterpriseConnection (see page 1305)

TDataSetProvider (see page 3145)

IAppServer

Writing Applications Using COM

33.1.1.1 TCOMConnection Members

The following tables list the members exposed by TCOMConnection.

Protected Methods

	Name	Description
	DoConnect (see page 1280)	Connects to an application server.
	DoDisconnect (see page 1292)	Drops a connection to an application server.
	GetConnected (see page 1293)	Returns the value of the Connected (see page 1296) property.
	GetServerCLSID (see page 1293)	Returns the GUID for the application server.
	Notification (see page 1294)	Responds when other objects are created or destroyed.
	SetConnected (see page 1280)	Sets the Connected (see page 1296) property.

Protected Properties

	Name	Description
	ObjectBroker (see page 1300)	Specifies an Object Broker that allows the connection component to choose from several application servers.

Public Methods

	Name	Description
	Create (see page 1291)	Creates an instance of a dispatch connection component.
	GetServer (see page 1293)	Returns the IAppServer interface of the application server.
	GetServerList (see page 1294)	Returns a list of remote data modules to which the component can connect.

Published Events

	Name	Description
	AfterConnect (see page 1302)	Occurs after a connection is established.
	AfterDisconnect (see page 1302)	Occurs after the connection closes.
	BeforeConnect (see page 1303)	Occurs immediately before establishing a connection.
	BeforeDisconnect (see page 1303)	Occurs immediately before the connection closes.
	OnGetUsername (see page 1304)	Occurs immediately before the remote login dialog.
	OnLogin (see page 1304)	Occurs when a communication channel opens to the server.

Published Properties

	Name	Description
	Connected (see page 1296)	Indicates whether the TDispatchConnection (see page 1290) object is connected to an application server.
	LoginPrompt (see page 1296)	Specifies whether a login dialog appears immediately before opening a new connection.
	ServerGUID (see page 1301)	Specifies the GUID of the application server with which to connect.
	ServerName (see page 1301)	Specifies the name of the application server with which to connect.

33.1.1.2 TCOMConnection Methods

The methods of the TCOMConnection class are listed here.

Protected Methods

	Name	Description
	DoConnect (see page 1280)	Connects to an application server.
	SetConnected (see page 1280)	Sets the Connected (see page 1296) property.

33.1.1.2.1 TCOMConnection.DoConnect Method

Connects to an application server.

Pascal

```
procedure DoConnect; override;
```

C++

```
virtual __fastcall DoConnect();
```

Description

DoConnect is called internally when the Connected (see page 1296) property is set to true. DoConnect creates a COM object for the application server (based on the ServerName (see page 1301) or ServerGUID (see page 1301) property), and sets the value of the AppServer property to this COM object.

Descendants of TCOMConnection (see page 1278) override DoConnect to change the way the client application forms a connection to an application server.

See Also

AfterConnect

BeforeConnect

Connected (see page 1296)

AppServer

ServerName (see page 1301)

ServerGUID (see page 1301)

33.1.1.2.2 TCOMConnection.SetConnected Method

Sets the Connected (see page 1296) property.

Pascal

```
procedure SetConnected(Value: Boolean); override;
```

C++

```
virtual __fastcall SetConnected(Boolean Value);
```

Description

SetConnected is the protected implementation of the Connected (see page 1296) property.

If Value is false, SetConnected generates a BeforeDisconnect (see page 1303) event, closes the connection, and generates an AfterDisconnect (see page 1302) event.

If Value is true, SetConnected performs the following actions:

1. It checks the value of the ServerGUID (see page 1301) or ServerName (see page 1301) property and raises an exception if these do not specify a valid application server.

2. If LoginPrompt (see page 1296) is true, and the connection is not already open, SetConnected generates an OnGetUsername (see page 1304) event, and uses that to initialize a login dialog. It displays the login dialog, and aborts the connection if the user cancels.
3. It generates a BeforeConnect (see page 1303) event.
4. It opens the connection.
5. It generates an AfterConnect (see page 1302) event.
6. If LoginPrompt (see page 1296) is true, and the connection opened successfully, SetConnected generates an OnLogin (see page 1304) event.

See Also

Connected (see page 1296)

ServerGUID (see page 1301)

ServerName (see page 1301)

BeforeConnect

AfterConnect

BeforeDisconnect

AfterDisconnect

LoginPrompt

OnGetUsername

OnLogin

33.1.2 TCustomObjectBroker Class

TCustomObjectBroker is the base class for components that locate available application servers for a DataSnap connection component.

Class Hierarchy

Pascal

```
TCustomObjectBroker = class(TComponent);
```

C++

```
class TCustomObjectBroker : public TComponent;
```

File

MConnect

Description

TCustomObjectBroker defines the methods by which a connection component obtains information about available application servers. When a connection component uses an object broker, it does not save any properties that identify a particular server machine. Instead, when it needs to open a connection, it uses the methods of the object broker to get that identifying information. This process allows client applications to locate server systems dynamically at runtime. If there are redundant server systems, this allows client applications to continue functioning when a server goes down.

TCustomObjectBroker does not implement any of the methods that provide information to connection components. Instead,

descendants such as TSimpleObjectBroker implement a mechanism for choosing servers and identifying them to connection components. Application developers can derive more elaborate object brokers that provide additional benefits such as intelligent load balancing.

See Also

TDispatchConnection (see page 1290)

TSimpleObjectBroker (see page 2475)

33.1.2.1 TCustomObjectBroker Members

The following tables list the members exposed by TCustomObjectBroker.

Public Methods

	Name	Description
	GetComputerForGUID (see page 1282)	Returns a string that identifies a server machine for an interface identified by its GUID.
	GetComputerForProgID (see page 1283)	Returns the machine name, URL, host name, or IP address of a server machine that supports an interface identified by its class name.
	GetPortForComputer (see page 1283)	Returns a port number for connecting to a specified remote server.
	SetConnectStatus (see page 1284)	Records whether a connection was established to a specified server.

33.1.2.2 TCustomObjectBroker Methods

The methods of the TCustomObjectBroker class are listed here.

Public Methods

	Name	Description
	GetComputerForGUID (see page 1282)	Returns a string that identifies a server machine for an interface identified by its GUID.
	GetComputerForProgID (see page 1283)	Returns the machine name, URL, host name, or IP address of a server machine that supports an interface identified by its class name.
	GetPortForComputer (see page 1283)	Returns a port number for connecting to a specified remote server.
	SetConnectStatus (see page 1284)	Records whether a connection was established to a specified server.

33.1.2.2.1 TCustomObjectBroker.GetComputerForGUID Method

Returns a string that identifies a server machine for an interface identified by its GUID.

Pascal

```
function GetComputerForGUID(GUID: TGUID): string; virtual; abstract;
```

C++

```
virtual __fastcall UnicodeString GetComputerForGUID(TGUID GUID) = 0;
```

Description

Call GetComputerForGUID to obtain a string that identifies a server machine on which a given interface is registered. The GUID parameter specifies the global unique identifier of the desired interface.

TCustomObjectBroker (see page 1281) declares GetComputerForGUID as an abstract (pure virtual) method. It is implemented in descendant objects that return a string designed to work with a particular connection component.

Connection components call this method, passing in the value of their ServerGUID property. They use the return value to set the ComputerName, URL, Host, or Address property, as appropriate.

See Also

GetComputerForProgID (see page 1283)

ServerGUID (see page 1301)

ComputerName (see page 1287)

Address

Host

URL

33.1.2.2.2 TCustomObjectBroker.GetComputerForProgID Method

Returns the machine name, URL, host name, or IP address of a server machine that supports an interface identified by its class name.

Pascal

```
function GetComputerForProgID(const ProgID): string; virtual; abstract;
```

C++

```
virtual __fastcall UnicodeString GetComputerForProgID(const  ProgID) = 0;
```

Description

Call GetComputerForProgID to obtain the name or address of a server machine on which a given interface is registered. The ProgID parameter specifies the name of the desired interface. If the server was created using one of the remote data module wizards, this is the value that was entered as the class name for the wizard.

TCustomObjectBroker (see page 1281) declares GetComputerForProgID as an abstract (pure virtual) method. It is implemented in descendant objects that return a string designed to work with a particular connection component.

Connection components can call this method, passing in the value of their ServerName property. They can use the return value to set the ComputerName, URL, Host, or Address property, as appropriate.

See Also

GetComputerForGUID (see page 1282)

ServerName (see page 1301)

ComputerName (see page 1287)

Address

Host

URL

33.1.2.2.3 TCustomObjectBroker.GetPortForComputer Method

Returns a port number for connecting to a specified remote server.

Pascal

```
function GetPortForComputer(const ComputerName: string): Integer; virtual; abstract;
```

C++

```
virtual __fastcall int GetPortForComputer(const UnicodeString ComputerName) = 0;
```

Description

Socket connection components call GetPortForComputer to get a value for their Port property. The ComputerName parameter specifies a Host name or IP address that was returned by GetComputerForGUID (see page 1282) or GetComputerForProgID (see page 1283).

None of the connection components that work on Linux explicitly use a Port number. GetPortForComputer is included in TCustomObjectBroker (see page 1281) to allow for the possibility of connection components that use an explicit Port number. The ComputerName parameter is a Host name or IP address that is returned by the GetComputerForGUID (see page 1282) or GetComputerForProgID (see page 1283) method.

TCustomObjectBroker (see page 1281) declares GetPortForComputer as an abstract (pure virtual) method. It is implemented in descendant objects that return a string designed to work with a particular connection component.

Note: This interface assumes that a single socket dispatcher on each server system handles all client requests, even when they are destined for different application servers on the same host.

See Also

GetComputerForProgID (see page 1283)

GetComputerForGUID (see page 1282)

Port

33.1.2.2.4 TCustomObjectBroker.SetConnectStatus Method

Records whether a connection was established to a specified server.

Pascal

```
procedure SetConnectStatus(ComputerName: string; Success: Boolean); virtual; abstract;
```

C++

```
virtual __fastcall SetConnectStatus(UnicodeString ComputerName, Boolean Success) = 0;
```

Description

After calling GetComputerForGUID (see page 1282) or GetComputerForProgID (see page 1283), connection components use the returned value to open a connection to the identified server. They then call SetConnectStatus to tell the broker whether the connection was successfully opened.

TCustomObjectBroker (see page 1281) declares SetConnectStatus as an abstract (pure virtual) method. Descendants override this method to keep track of when server machines are down or to monitor the number of connections created for each server.

See Also

GetComputerForGUID (see page 1282)

GetComputerForProgID (see page 1283)

33.1.3 TDCOMConnection Class

TDCOMConnection handles a client application's DCOM connection to the remote server in a in a multi-tiered application.

Class Hierarchy

Pascal

```
TDCOMConnection = class(TCOMConnection);
```

C++

```
class TDCOMConnection : public TCOMConnection;
```

File

MConnect

Description

TDCOMConnection is used in the client portion of a multi-tiered database application to establish and maintain the connection between the client and a remote application server. The DCOM connection object can

Establish an initial connection to a remote application server.

Obtain an IAppServer for the application server.

Obtain a list of providers on the application server.

Drop the connection to the remote application server.

TDCOMConnection uses DCOM to establish the initial connection between the client application and a remote application server. In order to do so, the application server must be registered on the client machine.

Once the connection is established, the client registers any or all of its client datasets with TDCOMConnection, and these client datasets use the IAppServer interface from the DCOM connection component to communicate with providers on the application server, or to otherwise call the application server's data module interface.

To connect client applications to remote application servers using TCP/IP sockets, use TSocketConnection instead. To connect to remote application servers using HTTP, use the TWebConnection component. To connect to application servers using SOAP, use TSoapConnection.

See Also

TClientDataSet

TSocketConnection

TRemoteDataModule

TDataSetProvider (see page 3145)

IAppServer

TWebConnection

TSoapConnection

Specifying a Connection Using DCOM

33.1.3.1 TDCOMConnection Members

The following tables list the members exposed by TDCOMConnection.

Protected Methods

	Name	Description
	DoConnect (see page 1287)	Connects to an application server.
	DoDisconnect (see page 1292)	Drops a connection to an application server.
	GetConnected (see page 1293)	Returns the value of the Connected (see page 1296) property.

	GetServerCLSID (see page 1293)	Returns the GUID for the application server.
	Notification (see page 1294)	Responds when other objects are created or destroyed.
	SetConnected (see page 1280)	Sets the Connected (see page 1296) property.
	SetConnected (see page 1295)	Sets the Connected (see page 1296) property.

Public Methods

	Name	Description
	Create (see page 1286)	Creates and initializes a DCOM connection component.
	GetServer (see page 1293)	Returns the IAppServer interface of the application server.
	GetServerList (see page 1294)	Returns a list of remote data modules to which the component can connect.

Published Events

	Name	Description
	AfterConnect (see page 1302)	Occurs after a connection is established.
	AfterDisconnect (see page 1302)	Occurs after the connection closes.
	BeforeConnect (see page 1303)	Occurs immediately before establishing a connection.
	BeforeDisconnect (see page 1303)	Occurs immediately before the connection closes.
	OnGetUsername (see page 1304)	Occurs immediately before the remote login dialog.
	OnLogin (see page 1304)	Occurs when a communication channel opens to the server.

Published Properties

	Name	Description
	ComputerName (see page 1287)	Specifies the name of the machine on which the application server is located.
	Connected (see page 1296)	Indicates whether the TDispatchConnection (see page 1290) object is connected to an application server.
	LoginPrompt (see page 1296)	Specifies whether a login dialog appears immediately before opening a new connection.
	ObjectBroker (see page 1288)	Specifies an Object Broker that allows the connection component to choose from several application servers.
	ServerGUID (see page 1301)	Specifies the GUID of the application server with which to connect.
	ServerName (see page 1301)	Specifies the name of the application server with which to connect.

33.1.3.2 TDCOMConnection.Create Constructor

Creates and initializes a DCOM connection component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TDCOMConnection(TComponent * AOwner);
```

Description

Call Create to instantiate TDCOMConnection (see page 1284) at runtime. DCOM connection components placed in forms or data modules at design time are created automatically.

The AOwner parameter specifies the component (usually the form or data module) that is responsible for freeing the DCOM connection component.

33.1.3.3 TDCOMConnection Methods

The methods of the TDCOMConnection class are listed here.

Protected Methods

	Name	Description
	DoConnect (see page 1287)	Connects to an application server.

33.1.3.3.1 TDCOMConnection.DoConnect Method

Connects to an application server.

Pascal

```
procedure DoConnect; override;
```

C++

```
virtual __fastcall DoConnect();
```

Description

DoConnect is called when the Connected (see page 1296) property is set to true. DoConnect establishes a dual interface with the remote data module on the application server. This interface is used to locate providers on the application server.

To locate the server system, DoConnect uses the ComputerName (see page 1287) property. If the ObjectBroker (see page 1288) property is set, DoConnect gets this value from the object broker. Otherwise, the application must assign this property a value directly.

See Also

AfterConnect

BeforeConnect

Connected (see page 1296)

AppServer

ComputerName (see page 1287)

ObjectBroker (see page 1300)

33.1.3.4 TDCOMConnection Properties

The properties of the TDCOMConnection class are listed here.

Published Properties

	Name	Description
	ComputerName (see page 1287)	Specifies the name of the machine on which the application server is located.
	ObjectBroker (see page 1288)	Specifies an Object Broker that allows the connection component to choose from several application servers.

33.1.3.4.1 TDCOMConnection.ComputerName Property

Specifies the name of the machine on which the application server is located.

Pascal

```
property ComputerName: string;
```

C++

```
__property UnicodeString ComputerName;
```

Description

Set ComputerName to the machine name of the computer where the application server resides. If ComputerName is blank (the default), TDCOMConnection (see page 1284) assumes the application server can be found on the user's machine or that the computer name can be derived from the system registry based on the DCOM configuration. If ComputerName is not blank, it overrides any system registry entries.

Normally the client and the application server run on different machines, so ComputerName must be set by the client application or available as part of the registry's DCOM configuration.

See Also

ServerName (see page 1301)

33.1.3.4.2 TDCOMConnection.ObjectBroker Property

Specifies an Object Broker that allows the connection component to choose from several application servers.

Pascal

```
property ObjectBroker: TCustomObjectBroker;
```

C++

```
__property TCustomObjectBroker ObjectBroker;
```

Description

Set ObjectBroker to use a broker that maintains a list of servers that support the application specified by ServerName or ServerGUID. When the connection component needs to open a connection, it requests information to identify a server machine from the Object Broker.

Descendants of TDispatchConnection define properties that specify how to locate server machines. These properties are not saved with the connection component if the ObjectBroker property is set. Instead, the Object Broker provides this information dynamically at runtime from its list of available servers.

Applications can use TSimpleObjectBroker for an object broker, or derive their own object brokers from TCustomObjectBroker.

See Also

DoConnect

ServerName (see page 1301)

ServerGUID (see page 1301)

TSimpleObjectBroker (see page 2475)

Brokering Connections

Specifying a Connection Using HTTP

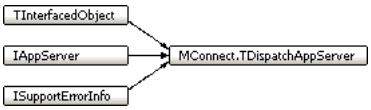
Specifying a Connection Using DCOM

Specifying a Connection Using Sockets

Using TSimpleObjectBroker

33.1.4 TDispatchAppServer Class

Class Hierarchy



Pascal

```
TDispatchAppServer = class(TInterfacedObject, IAppServer, ISupportErrorInfo);
```

C++

```
class TDispatchAppServer : public TInterfacedObject, public IAppServer, public
ISupportErrorInfo;
```

File

MConnect

Description

This is class MConnect.TDispatchAppServer.

33.1.4.1 TDispatchAppServer Members

The following tables list the members exposed by TDispatchAppServer.

Public Methods

	Name	Description
	Create (see page 1289)	This is Create, a member of class TDispatchAppServer.
	SafeCallException (see page 1290)	Handles exceptions in methods declared using the safecall calling convention.

33.1.4.2 TDispatchAppServer.Create Constructor

Pascal

```
constructor Create(const AppServer: IAppServerDisp);
```

C++

```
__fastcall TDispatchAppServer(const IAppServerDisp AppServer);
```

Description

This is Create, a member of class TDispatchAppServer.

33.1.4.3 TDispatchAppServer Methods

The methods of the TDispatchAppServer class are listed here.

Public Methods

	Name	Description
	SafeCallException (see page 1290)	Handles exceptions in methods declared using the safecall calling convention.

33.1.4.3.1 TDispatchAppServer.SafeCallException Method

Handles exceptions in methods declared using the safecall calling convention.

Pascal

```
function SafeCallException(ExceptObject: TObject; ExceptAddr: Pointer): HResult; override;
```

C++

```
virtual __fastcall HRESULT SafeCallException(TObject * ExceptObject, void * ExceptAddr);
```

Description

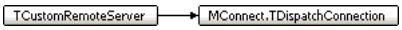
System::TObject::SafeCallException handles exceptions in methods that use the safecall calling convention. Some classes that implement interfaces override this method to handle errors that might occur.

As implemented in System::TObject, System::TObject::SafeCallException simply returns E_UNEXPECTED. This is the appropriate response for classes that do no support interfaces

33.1.5 TDispatchConnection Class

TDispatchConnection is the base class for objects that connect client applications to an IDispatch-based application server.

Class Hierarchy



Pascal

```
TDispatchConnection = class(TCustomRemoteServer);
```

C++

```
class TDispatchConnection : public TCustomRemoteServer;
```

File

MConnect

Description

Do not create instances of TDispatchConnection. Instead, use TDispatchConnection as a base class when creating a custom component that forms a connection to a COM-based application server in a multi-tiered database application.

TDispatchConnection introduces the infrastructure for logging on to an application server, obtaining an IAppServer interface, calling the application server's interface, and managing a remote connection. Additionally, TCustomRemoteServer inherits the ability to work with one or more client datasets.

TDispatchConnection is not capable of connecting to an application server. This ability is added by descendants, which are specialized for different connection types. Once a connection is formed, however, TDispatchConnection makes the application server's IAppServer interface available to TClientDataSet components.

To enable a client application to connect to a COM-based application server, use a descendant of TDispatchConnection such as TSocketConnection, TWebConnection, or TDCOMConnection (see page 1284).

See Also

TDataSetProvider (see page 3145)

TClientDataSet

TDCOMConnection (see page 1284)

TSocketConnection

IAppServer

TWebConnection

TSoapConnection

33.1.5.1 TDispatchConnection Members

The following tables list the members exposed by TDispatchConnection.

Protected Methods

	Name	Description
	DoDisconnect (see page 1292)	Drops a connection to an application server.
	GetConnected (see page 1293)	Returns the value of the Connected (see page 1296) property.
	GetServerCLSID (see page 1293)	Returns the GUID for the application server.
	Notification (see page 1294)	Responds when other objects are created or destroyed.
	SetConnected (see page 1295)	Sets the Connected (see page 1296) property.

Protected Properties

	Name	Description
	ObjectBroker (see page 1300)	Specifies an Object Broker that allows the connection component to choose from several application servers.

Public Methods

	Name	Description
	Create (see page 1291)	Creates an instance of a dispatch connection component.
	GetServer (see page 1293)	Returns the IAppServer interface of the application server.
	GetServerList (see page 1294)	Returns a list of remote data modules to which the component can connect.

Published Events

	Name	Description
	AfterConnect (see page 1302)	Occurs after a connection is established.
	AfterDisconnect (see page 1302)	Occurs after the connection closes.
	BeforeConnect (see page 1303)	Occurs immediately before establishing a connection.
	BeforeDisconnect (see page 1303)	Occurs immediately before the connection closes.
	OnGetUsername (see page 1304)	Occurs immediately before the remote login dialog.
	OnLogin (see page 1304)	Occurs when a communication channel opens to the server.

Published Properties

	Name	Description
	Connected (see page 1296)	Indicates whether the TDispatchConnection (see page 1290) object is connected to an application server.
	LoginPrompt (see page 1296)	Specifies whether a login dialog appears immediately before opening a new connection.
	ServerGUID (see page 1301)	Specifies the GUID of the application server with which to connect.
	ServerName (see page 1301)	Specifies the name of the application server with which to connect.

33.1.5.2 TDispatchConnection.Create Constructor

Creates an instance of a dispatch connection component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TDispatchConnection(TComponent * AOwner);
```

Description

Call Create to instantiate a dispatch connection component at runtime. Connection components placed in forms or data modules at design time are created automatically.

After calling the inherited constructor, Create initializes the LoginPrompt (see page 1296) property to false.

See Also

LoginPrompt

Create

33.1.5.3 TDispatchConnection Methods

The methods of the TDispatchConnection class are listed here.

Protected Methods

	Name	Description
	DoDisconnect (see page 1292)	Drops a connection to an application server.
	GetConnected (see page 1293)	Returns the value of the Connected (see page 1296) property.
	GetServerCLSID (see page 1293)	Returns the GUID for the application server.
	Notification (see page 1294)	Responds when other objects are created or destroyed.
	SetConnected (see page 1295)	Sets the Connected (see page 1296) property.

Public Methods

	Name	Description
	GetServer (see page 1293)	Returns the IAppServer interface of the application server.
	GetServerList (see page 1294)	Returns a list of remote data modules to which the component can connect.

33.1.5.3.1 TDispatchConnection.DoDisconnect Method

Drops a connection to an application server.

Pascal

```
procedure DoDisconnect; override;
```

C++

```
virtual __fastcall DoDisconnect();
```

Description

DoDisconnect is called internally when the Connected (see page 1296) property is set to false. It drops the connection and changes the AppServer property to a NULL Variant.

See Also

AfterDisconnect

BeforeDisconnect

Connected (see page 1296)

AppServer

33.1.5.3.2 TDispatchConnection.GetConnected Method

Returns the value of the Connected (see page 1296) property.

Pascal

```
function GetConnected: Boolean; override;
```

C++

```
virtual __fastcall Boolean GetConnected();
```

Description

GetConnected is the protected implementation of the Connected (see page 1296) property. It uses the AppServer property (set by the DoConnect method of a TDispatchConnection (see page 1290) descendant) to determine whether a connection is established.

If a connection is established, GetConnected returns true. Otherwise, GetConnected returns false.

See Also

Connected (see page 1296)

DoConnect (see page 1280)

33.1.5.3.3 TDispatchConnection.GetServer Method

Returns the IAppServer interface of the application server.

Pascal

```
function GetServer: IAppServer; override;
```

C++

```
virtual __fastcall IAppServer GetServer();
```

Description

Client datasets call GetServer to set the value of their AppServer property. This allows them to access providers on the remote application server.

GetServer ensures that there is a connection and then returns an IAppServer interface that can be used to communicate with providers on the application server. To use other methods of the application server's interface, use the AppServer property instead.

See Also

Connected (see page 1296)

AppServer

AppServer

33.1.5.3.4 TDispatchConnection.GetServerCLSID Method

Returns the GUID for the application server.

Pascal

```
function GetServerCLSID: TGUID;
```

C++

```
__fastcall TGUID GetServerCLSID();
```

Description

GetServerCLSID is used internally to obtain the GUID for the application server.

In most cases, GetServerCLSID returns the value of the ServerGUID (see page 1301) property. However, if the application server was specified using the ServerName (see page 1301) property, and the application server was not registered when the ServerName (see page 1301) property was set, ServerGUID (see page 1301) will not represent the application server's GUID. GetServerCLSID can return a GUID that only becomes available when a connection is opened.

If neither the ServerName (see page 1301) nor the ServerGUID (see page 1301) property is set, GetServerCLSID raises an exception.

See Also

ServerGUID (see page 1301)

ServerName (see page 1301)

33.1.5.3.5 TDispatchConnection.GetServerList Method

Returns a list of remote data modules to which the component can connect.

Pascal

```
function GetServerList: OleVariant; override;
```

C++

```
virtual __fastcall OleVariant GetServerList();
```

Description

Connection components use GetServerList internally to fill the drop down list in the Object Inspector for the ServerName (see page 1301) property. GetServerList returns a variant array that lists all application servers that are on the local machine, registered in the system registry, or on the remote machine identified by the connection component.

See Also

ServerName (see page 1301)

33.1.5.3.6 TDispatchConnection.Notification Method

Responds when other objects are created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification is called automatically so objects can make internal adjustments when components are created or destroyed. TDispatchConnection (see page 1290) overrides the inherited method to set the ObjectBroker (see page 1300) property to nil (Delphi) or NULL (C++) when it receives notification that that object is about to be destroyed.

See Also

ObjectBroker (see page 1300)

33.1.5.3.7 TDispatchConnection.SetConnected Method

Sets the Connected (see page 1296) property.

Pascal

```
procedure SetConnected(Value: Boolean); override;
```

C++

```
virtual __fastcall SetConnected(Boolean Value);
```

Description

SetConnected is the protected implementation of the Connected (see page 1296) property. It establishes a connection to the Application server if Value is true, and disconnects from the application server if Value is false.

If Value is false, SetConnected generates a BeforeDisconnect (see page 1303) event, closes the connection, and generates an AfterDisconnect (see page 1302) event.

If Value is true, SetConnected performs the following actions:

1. If LoginPrompt (see page 1296) is true, and the connection is not already open, SetConnected generates an OnGetUsername (see page 1304) event, and uses that to initialize a login dialog. It displays the login dialog, and aborts the connection if the user cancels.
2. It generates a BeforeConnect (see page 1303) event.
3. It opens the connection.
4. It generates an AfterConnect (see page 1302) event.
5. If LoginPrompt (see page 1296) is true, and the connection opened successfully, SetConnected generates an OnLogin (see page 1304) event.

See Also

Connected (see page 1296)

BeforeConnect

AfterConnect

BeforeDisconnect

AfterDisconnect

LoginPrompt

OnGetUsername

OnLogin

33.1.5.4 TDispatchConnection Properties

The properties of the TDispatchConnection class are listed here.

Protected Properties

	Name	Description
	ObjectBroker (see page 1300)	Specifies an Object Broker that allows the connection component to choose from several application servers.

Published Properties

	Name	Description
	Connected (see page 1296)	Indicates whether the TDispatchConnection (see page 1290) object is connected to an application server.
	LoginPrompt (see page 1296)	Specifies whether a login dialog appears immediately before opening a new connection.
	ServerGUID (see page 1301)	Specifies the GUID of the application server with which to connect.
	ServerName (see page 1301)	Specifies the name of the application server with which to connect.

33.1.5.4.1 TDispatchConnection.Connected Property

Indicates whether the TDispatchConnection (see page 1290) object is connected to an application server.

Pascal

```
property Connected;
```

C++

```
__property Connected;
```

Description

Set Connected to true to establish a connection to the application server. Set Connected to false to terminate the connection.

When changing the value of Connected, TDispatchConnection (see page 1290) generates a BeforeConnect (see page 1303) and an AfterConnect (see page 1302) event, or a BeforeDisconnect (see page 1303) and an AfterDisconnect (see page 1302) event. In addition, when setting Connected to true, TDispatchConnection (see page 1290) may display a login dialog, depending on the value of LoginPrompt (see page 1296).

The Connected property uses the protected DoConnect and DoDisconnect (see page 1292) methods to connect to or disconnect from an application server. Descendants of TDispatchConnection (see page 1290) override DoConnect and DoDisconnect (see page 1292) to change the way the connection is formed or terminated.

See Also

DoDisconnect (see page 1292)

LoginPrompt

AfterConnect

AfterDisconnect

BeforeConnect

BeforeDisconnect

DoConnect (see page 1280)

33.1.5.4.2 TDispatchConnection.LoginPrompt Property

Specifies whether a login dialog appears immediately before opening a new connection.

Pascal

```
property LoginPrompt;
```

C++

```
__property LoginPrompt;
```

Description

Set LoginPrompt to true to provide login support when establishing a connection. LoginPrompt controls two things: the occurrence of the OnLogin event, and the appearance of a default login dialog that prompts users for a name and password when you include DBLogDlg.hpp in your unit (C++) or add DBLogDlg to your uses clause (Delphi). When the OnLogin event occurs and when the default login dialog appears depend on the type of connection component:

For TDatabase, TIBDatabase, and TSQLConnection, the dialog appears after the BeforeConnect event and before the AfterConnect event, unless you supply an OnLogin event handler. If there is an OnLogin event handler, that event occurs in place of the login dialog, and there is no need to include DBLogDlg.hpp in your unit (C++) or add DBLogDlg to your uses clause (Delphi). If correct values for the user name and password are not supplied in the dialog or by the OnLogin event handler, the connection fails. The OnLogin event does not fire unless LoginPrompt is set to true.

For TADOConnection components, the dialog appears after the OnWillConnect event and before the BeforeConnect event. If there is an OnLogin event handler, that event occurs after the login dialog. If you do not include DBLogDlg.hpp in your unit (C++) or add DBLogDlg to your uses clause (Delphi), the OnLogin event occurs but there is no default login dialog. If correct values for the user name and password are not supplied in the dialog or by the OnLogin event handler, the connection fails. The OnLogin event does not fire unless LoginPrompt is set to true.

For DataSnap SOAP connection components, the dialog appears after the OnGetUsername event and before the BeforeConnect, AfterConnect, and OnLogin events. If the user cancels from the login dialog, no attempt is made to open a connection.

When LoginPrompt is false, the application must supply user name and password values programmatically:

For TDatabase, the user name and password can be supplied as USER_NAME and PASSWORD parameters in the Params property.

For TADOConnection, the user name and password can be supplied as the ConnectionString property.

For TSQLConnection, the user name, password, and database can be supplied as UserName, Password, and Database parameters in the Params property, or provided as connection parameters associated with the connection name.

For DataSnap SOAP connection components, there is no built-in use for the user name and password supplied by the login dialog. (The UserName and Password properties of TWebConnection are unrelated).

Warning: Storing hard-coded user name and password entries as property values or in code for an OnLogin event handler can compromise server security.

C++ Examples:

```
//
//This example demostrates the use of ADO for database conectivity.
//Example assumes that a TDBGrid is placed on the form.
//
void __fastcall TForm2::FormCreate(TObject *Sender)
{
    /* Login details */
    String UserName = "db_user_name";
    String PassWord = "db_pass_word";
    String Server = "my.db.server";

    /* Connection String */
    String ConnString =
        "Provider=SQLOLEDB.1;Persist Security Info=False;";
    ConnString +=
        "User ID=%s;Password=%s;Data Source=%s;Use Procedure for Prepare=1;";
    ConnString +=
        "Auto Translate=True;Packet Size=4096;Use Encryption for Data=False;";
    ConnString +=
```

```
        "Tag with column collation when possible=False";

    /* SQL Query */
    String SQLStr = "SELECT * FROM customer WHERE customer_id = :AnId;";

    /* All ADO variables */
    TADOConnection* ADOConn;
    TADOQuery* ADOQuery;
    TDataSource* DataSrc;
    TParameter* Param;

    /* Create an ADO connection */
    ADOConn = new TADOConnection(this);

    /* Setup the connection string */
    ADOConn->ConnectionString = Format(ConnString,
        ARRAYOFCONST((UserName, PassWord, Server)));

    /* Disable login prompt */
    ADOConn->LoginPrompt = False;

    try
    {
        ADOConn->Connected = true;
    }
    catch (EADOError *e)
    {
        MessageDlg("Error while connecting", mtError,
                   TMsgDlgButtons() << mbOK, 0);
        return;
    }

    /* Create the query */
    ADOQuery = new TADOQuery(this);
    ADOQuery->Connection = ADOConn;
    ADOQuery->SQL->Add(SQLStr);

    /* Update the parameter that was parsed from the SQL query: AnId */
    Param = ADOQuery->Parameters->ParamByName("AnId");
    Param->DataType = ftInteger;
    Param->Value = 1;

    /* Set the query to Prepared - will improve performance */
    ADOQuery->Prepared = true;

    try
    {
        ADOQuery->Active = true;
    }
    catch (EADOError *e)
    {
        MessageDlg("Error while connecting", mtError,
                   TMsgDlgButtons() << mbOK, 0);
        return;
    }

    /* Create the data source */
    DataSrc = new TDataSource(this);
    DataSrc->DataSet = ADOQuery;
    DataSrc->Enabled = true;

    /* Finally initilalize the grid */
    DBGrid1->DataSource = DataSrc;
}
```

Delphi Examples:

```
{
This example demostrates the use of ADO for database conectivity.
Example assumes that a TDBGrid is placed on the form.
}
procedure TForm2.FormCreate(Sender: TObject);
const
  { Connection string }
  ConnString =
  'Provider=SQLOLEDB.1;Persist Security Info=False;' +
  'User ID=%s;Password=%s;Data Source=%s;Use Procedure for Prepare=1;' +
  'Auto Translate=True;Packet Size=4096;Use Encryption for Data=False;'+
  'Tag with column collation when possible=False';

  { SQL Query }
  SQLStr = 'SELECT * FROM customer WHERE customer_id = :AnId;';

  { User access }
  UserName = 'db_user_name';
  PassWord = 'db_pass_word';
  Server = 'my.db.server';

var
  ADOConn  : TADOConnection;
  ADOQuery : TADOQuery;
  DataSrc  : TDataSource;
  Param    : TParameter;

begin
  { Create an ADO connection }
  ADOConn := TADOConnection.Create(Self);
  { Setup the provider engine }

  { Setup the connection string }
  ADOConn.ConnectionString := Format(ConnString,
    [UserName, PassWord, Server]);

  { Disable login prompt }
  ADOConn.LoginPrompt := False;

  try
    ADOConn.Connected := True;
  except
    on e: EADOError do
    begin
      MessageDlg('Error while connecting', mtError,
                 [mbOK], 0);

      Exit;
    end;
  end;

  { Create the query }
  ADOQuery := TADOQuery.Create(Self);
  ADOQuery.Connection := ADOConn;
  ADOQuery.SQL.Add(SQLStr);

  { Update the parameter that was parsed from the SQL query: AnId }
  Param := ADOQuery.Parameters.ParamByName('AnId');
  Param.DataType := ftInteger;
  Param.Value := 1;

  { Set the query to Prepared - will improve performance }
  ADOQuery.Prepared := true;

  try
    ADOQuery.Active := True;
```

```
    except
      on e: EADOError do
      begin
        MessageDlg('Error while doing query', mtError,
                   [mbOK], 0);

        Exit;
      end;
    end;

    { Create the data source }
    DataSrc := TDataSource.Create(Self);
    DataSrc.DataSet := ADOQuery;
    DataSrc.Enabled := true;

    { Finally initilalize the grid }
    DBGrid1.DataSource := DataSrc;
  end;
```

See Also

OnLogin

BeforeConnect

AfterConnect

OnGetUsername

OnLogin

OnLogin

Controlling Server Login

33.1.5.4.3 TDispatchConnection.ObjectBroker Property

Specifies an Object Broker that allows the connection component to choose from several application servers.

Pascal

```
property ObjectBroker: TCustomObjectBroker;
```

C++

```
__property TCustomObjectBroker ObjectBroker;
```

Description

Set ObjectBroker to use a broker that maintains a list of servers that support the application specified by ServerName (see page 1301) or ServerGUID (see page 1301). When the connection component needs to open a connection, it requests information to identify a server machine from the Object Broker.

Descendants of TDispatchConnection (see page 1290) define properties that specify how to locate server machines. These properties are not saved with the connection component if the ObjectBroker property is set. Instead, the Object Broker provides this information dynamically at runtime from its list of available servers.

Applications can use TSimpleObjectBroker for an object broker, or derive their own object brokers from TCustomObjectBroker (see page 1281).

See Also

DoConnect

ServerName (see page 1301)

ServerGUID (see page 1301)

TSimpleObjectBroker (see page 2475)

Brokering Connections

Specifying a Connection Using HTTP

Specifying a Connection Using DCOM

Specifying a Connection Using Sockets

Using TSimpleObjectBroker

33.1.5.4.4 TDispatchConnection.ServerGUID Property

Specifies the GUID of the application server with which to connect.

Pascal

```
property ServerGUID: string;
```

C++

```
__property UnicodeString ServerGUID;
```

Description

Use ServerGUID to specify the application server to which the client application should connect by its GUID (global unique identifier). If the value specifies a valid GUID for a registered application server, setting ServerGUID changes the value of ServerName (see page 1301) to the class name of the corresponding COM object.

Using ServerGUID rather than ServerName (see page 1301) to identify the application server is more robust because it does not require the application server to be registered on the client system.

Note: Either ServerName (see page 1301) or ServerGUID must be provided so that the dispatch connection can create and communicate with the appropriate server COM object. This is true even for connections that are not formed using DCOM, because the remote data module on the application server is always implemented as a COM object.

See Also

ServerName (see page 1301)

Connecting to the Application Server

Connecting to the Application Server using DataSnap Components

33.1.5.4.5 TDispatchConnection.ServerName Property

Specifies the name of the application server with which to connect.

Pascal

```
property ServerName: string;
```

C++

```
__property UnicodeString ServerName;
```

Description

Use ServerName to specify the name of the application server to which the client application should connect. At design time, the Object Inspector provides a drop-down list of all available servers if one of the following conditions is met:

For DCOM and OLEnterprise connections, the servers must be registered in the system Registry (see page 3235).

For Socket connections, ScktSrvr.exe must be running on the server machine.

For Web connections, the connection component must have an URL for locating httpsrvr.dll.

If ServerName specifies a valid name for a registered application server, setting ServerName sets the ServerGUID (see page 1301) property to the GUID that corresponds to the named application server.

Note: Either ServerName or ServerGUID (see page 1301) must be provided so that the dispatch connection can create and communicate with the appropriate server COM object. This is true even for connections that are not formed using DCOM, because the remote data module on the application server is always implemented as a COM object. Using ServerGUID (see page 1301) rather than ServerName to identify the application server is more robust because it does not require the application server to be registered on the client system when using a COM-based connection.

See Also

ServerGUID (see page 1301)

Connecting to the Application Server

33.1.5.5 TDispatchConnection Events

The events of the TDispatchConnection class are listed here.

Published Events

	Name	Description
	AfterConnect (see page 1302)	Occurs after a connection is established.
	AfterDisconnect (see page 1302)	Occurs after the connection closes.
	BeforeConnect (see page 1303)	Occurs immediately before establishing a connection.
	BeforeDisconnect (see page 1303)	Occurs immediately before the connection closes.
	OnGetUsername (see page 1304)	Occurs immediately before the remote login dialog.
	OnLogin (see page 1304)	Occurs when a communication channel opens to the server.

33.1.5.5.1 TDispatchConnection.AfterConnect Event

Occurs after a connection is established.

Pascal

```
property AfterConnect;
```

C++

```
__property AfterConnect;
```

Description

Write an AfterConnect event handler to take application-specific actions immediately after the connection component opens a connection to the remote source of database information.

See Also

BeforeConnect

Connected

Connecting to a Database Server

33.1.5.5.2 TDispatchConnection.AfterDisconnect Event

Occurs after the connection closes.

Pascal

```
property AfterDisconnect;
```

C++

```
__property AfterDisconnect;
```

Description

Write an AfterDisconnect event handler to take application-specific actions after the connection component drops a connection.

See Also

BeforeDisconnect

Connected

Disconnecting from a Database Server

33.1.5.5.3 TDispatchConnection.BeforeConnect Event

Occurs immediately before establishing a connection.

Pascal

```
property BeforeConnect;
```

C++

```
__property BeforeConnect;
```

Description

Write a BeforeConnect event handler to take application-specific actions before the connection component opens a connection to the remote source of database information.

See Also

AfterConnect

Connected

Connecting to a Database Server

33.1.5.5.4 TDispatchConnection.BeforeDisconnect Event

Occurs immediately before the connection closes.

Pascal

```
property BeforeDisconnect;
```

C++

```
__property BeforeDisconnect;
```

Description

Write a BeforeDisconnect event handler to take application-specific actions before dropping a connection.

Delphi Examples:

```
{
This BeforeDisconnect event handler makes sure that the
client dataset has applied all its pending updates before
terminating the connection to the server.
}
```

```
procedure TForm1.RemoteServer1BeforeDisconnect(Sender: TObject);
begin
  if (ClientDataSet1.ChangeCount > 0) then
    ClientDataSet1.ApplyUpdates(-1);
end;
```

See Also

AfterDisconnect

Connected

Disconnecting from a Database Server

33.1.5.5.5 TDispatchConnection.OnGetUsername Event

Occurs immediately before the remote login dialog.

Pascal

```
property OnGetUsername;
```

C++

```
__property OnGetUsername;
```

Description

Write an OnGetUsername event handler to initialize the user name in the remote login dialog that appears when the LoginPrompt property is true. Set the Username parameter to provide the initial value for the user name. If no OnGetUsername event handler is provided, the user name is initialized to an empty string.

The remote login dialog occurs immediately before a new connection is established to the application server. If the user cancels from that dialog, the connection attempt is aborted.

Note: The OnGetUsername event does not occur unless the LoginPrompt property is true.

See Also

LoginPrompt

OnLogin

33.1.5.5.6 TDispatchConnection.OnLogin Event

Occurs when a communication channel opens to the server.

Pascal

```
property OnLogin;
```

C++

```
__property OnLogin;
```

Description

OnLogin occurs when connections are opened if the LoginPrompt property is true.

For DataSnap SOAP connection components, the following sequence of events occurs when opening a connection to an application server:

1. An OnGetUsername event occurs, where the application can supply an initial user name for the login dialog.
2. The login dialog appears, where the user can supply a user name and password.

3. A BeforeConnect event occurs, the connection is opened, and an AfterConnect event occurs.
4. The OnLogin event occurs, where you can write code to log in to the application server using the username and password that were provided in the remote login dialog.

   Note: Do not use the OnLogin event if the connection component provides a different scheme for logging in. For example, TWebConnection uses the Username and Password properties instead.

   For ADO connection components, the following sequence of events occurs when opening a connection to an application server:

1. A BeforeConnect event occurs.

2. An OnWillConnect event occurs, where the application can supply an initial user name and password for the login dialog, as well as the connection options and status.

3. The login dialog appears, where the user can supply a user name and password.

4. The OnLogin event occurs, where you can write code to respond to the user-supplied values. (For example, you could convert an application-specific user name and password to database user name and password).

5. The connection is opened, and an AfterConnect event occurs.

Note: Some

C connection components that do not connect to an application server in a multitiered database application replace the OnLogin event defined in TCustomConnection with a different event that occurs before a connection is established.

See Also

Connected

Open

OnGetUsername

BeforeConnect

AfterConnect

LoginPrompt

OnWillConnect

33.1.6 TOLEnterpriseConnection Class

TOLEnterpriseConnection is provided for backward compatibility with applications that connect to an application server using OLEnterprise.

Class Hierarchy

Pascal

```
TOLEnterpriseConnection = class(TCOMConnection);
```

C++

```
class TOLEnterpriseConnection : public TCOMConnection;
```

File

MConnect

Description

TOLEnterpriseConnection is used in the client portion of a multi-tiered database application to establish and maintain the connection between the client and a remote application server using OLEnterprise. This component is provided for backward compatibility.

TOLEnterpriseConnection establishes the initial connection between the client application and a remote application server using OLEnterprise. Once the connection is established, the client registers any or all of its client datasets with the remote server, and these client datasets use the IAppServer interface from the OLEnterprise connection component to communicate with providers on the application server, or to otherwise call the application server's data module interface.

To connect to remote application servers using DCOM, use the TDCOMConnection (see page 1284) component instead. To connect client applications to remote application servers using TCP/IP sockets, use TSocketConnection. To connect client applications to remote application servers using HTTP, use TWebConnection. To connect to remote application servers using SOAP, use TSOAPConnection.

See Also

TClientDataSet

TSocketConnection

TDCOMConnection (see page 1284)

TRemoteDataModule

IAppServer

TWebConnection

33.1.6.1 TOLEnterpriseConnection Members

The following tables list the members exposed by TOLEnterpriseConnection.

Protected Methods

	Name	Description
	DoConnect (see page 1307)	Establishes a connection to the application server.
	DoDisconnect (see page 1292)	Drops a connection to an application server.
	GetConnected (see page 1293)	Returns the value of the Connected (see page 1296) property.
	GetServerCLSID (see page 1293)	Returns the GUID for the application server.
	Notification (see page 1294)	Responds when other objects are created or destroyed.
	SetConnected (see page 1280)	Sets the Connected (see page 1296) property.
	SetConnected (see page 1295)	Sets the Connected (see page 1296) property.

Protected Properties

	Name	Description
	ObjectBroker (see page 1300)	Specifies an Object Broker that allows the connection component to choose from several application servers.

Public Methods

	Name	Description
	Create (see page 1291)	Creates an instance of a dispatch connection component.
	GetServer (see page 1293)	Returns the IAppServer interface of the application server.
	GetServerList (see page 1294)	Returns a list of remote data modules to which the component can connect.

Published Events

	Name	Description
	AfterConnect (see page 1302)	Occurs after a connection is established.
	AfterDisconnect (see page 1302)	Occurs after the connection closes.
	BeforeConnect (see page 1303)	Occurs immediately before establishing a connection.
	BeforeDisconnect (see page 1303)	Occurs immediately before the connection closes.
	OnGetUsername (see page 1304)	Occurs immediately before the remote login dialog.
	OnLogin (see page 1304)	Occurs when a communication channel opens to the server.

Published Properties

	Name	Description
	BrokerName (see page 1308)	Specifies the name of the Business Object Broker that locates the application server.
	ComputerName (see page 1308)	Specifies the name of the machine on which the application server is located.
	Connected (see page 1296)	Indicates whether the TDispatchConnection (see page 1290) object is connected to an application server.
	LoginPrompt (see page 1296)	Specifies whether a login dialog appears immediately before opening a new connection.
	ServerGUID (see page 1301)	Specifies the GUID of the application server with which to connect.
	ServerName (see page 1301)	Specifies the name of the application server with which to connect.

33.1.6.2 TOLEnterpriseConnection Methods

The methods of the TOLEnterpriseConnection class are listed here.

Protected Methods

	Name	Description
	DoConnect (see page 1307)	Establishes a connection to the application server.

33.1.6.2.1 TOLEnterpriseConnection.DoConnect Method

Establishes a connection to the application server.

Pascal

```
procedure DoConnect; override;
```

C++

```
virtual __fastcall DoConnect();
```

Description

DoConnect is called when the Connected (see page 1296) property is set to true. DoConnect obtains the interface of the remote data module on the application server using OLEnterprise runtime. This interface can then be used to obtain interfaces for the providers on the application server or to make interface calls to the remote data module.

See Also

AfterConnect

BeforeConnect

Connected (see page 1296)

AppServer

33.1.6.3 TOLEnterpriseConnection Properties

The properties of the TOLEnterpriseConnection class are listed here.

Published Properties

	Name	Description
	BrokerName (see page 1308)	Specifies the name of the Business Object Broker that locates the application server.
	ComputerName (see page 1308)	Specifies the name of the machine on which the application server is located.

33.1.6.3.1 TOLEnterpriseConnection.BrokerName Property

Specifies the name of the Business Object Broker that locates the application server.

Pascal

```
property BrokerName: string;
```

C++

```
__property UnicodeString BrokerName;
```

Description

Use BrokerName to specify the OLEnterprise Business Object Broker with which the application server is registered. The Business Object Broker locates an available application server with the name specified by the ServerName (see page 1301) property.

If you are not using the Business Object Broker, use the ComputerName (see page 1308) property instead. You must specify a value for either BrokerName or ComputerName (see page 1308):

Note: Setting the value of BrokerName changes the ComputerName (see page 1308) property to blank, and vice versa.

See Also

ServerName (see page 1301)

ComputerName (see page 1308)

33.1.6.3.2 TOLEnterpriseConnection.ComputerName Property

Specifies the name of the machine on which the application server is located.

Pascal

```
property ComputerName: string;
```

C++

```
__property UnicodeString ComputerName;
```

Description

Use ComputerName to specify the machine where the application server resides if you are not using the Business Object Broker. If you are not using the Business Object Broker and ComputerName is blank (the default), TOLEnterpriseConnection (see page 1305) assumes the application server can be found on the user's machine or that the computer name can be derived from the system registry based on the DCOM configuration. If ComputerName is not blank, it overrides any system registry entries.

If you are using the Business Object Broker, use the BrokerName (see page 1308) property instead. BrokerName (see page 1308) and ComputerName are mutually exclusive: Setting the value of ComputerName changes the BrokerName (see page

1308) property to blank, and vice versa.

See Also

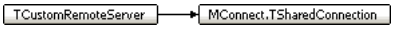
ServerName (see page 1301)

BrokerName (see page 1308)

33.1.7 TSharedConnection Class

TSharedConnection connects to a child remote data module that is accessible only through the properties of a main remote data module on the application server.

Class Hierarchy



Pascal

```
TSharedConnection = class(TCustomRemoteServer);
```

C++

```
class TSharedConnection : public TCustomRemoteServer;
```

File

MConnect

Description

TSharedConnection manages a connection to a "child" remote data module that is managed by a single main remote data module on an application server. The client application does not connect directly to the "child" remote data module. Rather, it uses a single connection to the main remote data module on the server machine, and that main remote data module dispatches calls to the IAppServer interface of other "child" data modules. This allows the client application to use a single connection rather than one connection for each remote data module on the application server.

On the server-side, the interface for the child remote data module is surfaced as a property of the main remote data module's interface. The TSharedConnection component uses this property name to indicate which child remote data module is the actual target of its connection.

TSharedConnection uses a TDispatchConnection (see page 1290) descendant (TDCOMConnection (see page 1284), TWebConnection, or TSocketConnection) to form the connection to the main remote data module on the application server. When you use TSharedConnection to open a connection, it causes the associated dispatch connection to connect to the main remote data module if it has not already done so. When you close the connection maintained by TSharedConnection, however, it severs only the connection to the child remote data module, leaving the connection to the main remote data module open in case other components on the client application are still using it.

See Also

TCustomClientDataSet

TSocketConnection

TWebConnection

TDCOMConnection (see page 1284)

Connecting to an Application Server That Uses Multiple Data Modules

33.1.7.1 TSharedConnection Members

The following tables list the members exposed by TSharedConnection.

Protected Methods

	Name	Description
	ConnectEvent (see page 1311)	This is ConnectEvent, a member of class TSharedConnection.
	DoConnect (see page 1311)	Provides the interface for a method that opens a connection.
	DoDisconnect (see page 1312)	Provides the interface for a method that terminates the connection.
	GetConnected (see page 1312)	Returns the value of the Connected property.
	Notification (see page 1313)	Forwards notification messages to all owned components.

Public Methods

	Name	Description
	Destroy (see page 1311)	Destroys the TSharedConnection (see page 1309) instance.
	GetServer (see page 1313)	Returns the IAppServer interface of the child remote data module.

Published Events

	Name	Description
	AfterConnect (see page 1315)	Occurs after a connection is established.
	AfterDisconnect (see page 1315)	Occurs after the connection closes.
	BeforeConnect (see page 1315)	Occurs immediately before establishing a connection.
	BeforeDisconnect (see page 1316)	Occurs immediately before the connection closes.

Published Methods

	Name	Description
	Create (see page 1310)	Creates an instance of a remote server component.

Published Properties

	Name	Description
	ChildName (see page 1314)	Identifies the child remote data module on the application server with which you want to communicate.
	ParentConnection (see page 1314)	Specifies the connection component that establishes a connection to the main remote data module on the application server.

33.1.7.2 TSharedConnection.Create Constructor

Creates an instance of a remote server component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TSharedConnection(TComponent * AOwner);
```

Description

Create instantiates an instance of a remote server component and initializes its properties.

AOwner is the component that is responsible for freeing the TCustomRemoteServer instance. It becomes the value of the Owner property.

See Also

Create

Owner

33.1.7.3 TSharedConnection.Destroy Destructor

Destroys the TSharedConnection (see page 1309) instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TSharedConnection();
```

Description

Do not call Destroy directly in an application. Instead, call Free, which verifies that the reference to the shared connection component is not nil before calling Destroy

See Also

System::TObject::Free

33.1.7.4 TSharedConnection Methods

The methods of the TSharedConnection class are listed here.

Protected Methods

	Name	Description
	ConnectEvent (see page 1311)	This is ConnectEvent, a member of class TSharedConnection.
	DoConnect (see page 1311)	Provides the interface for a method that opens a connection.
	DoDisconnect (see page 1312)	Provides the interface for a method that terminates the connection.
	GetConnected (see page 1312)	Returns the value of the Connected property.
	Notification (see page 1313)	Forwards notification messages to all owned components.

Public Methods

	Name	Description
	GetServer (see page 1313)	Returns the IAppServer interface of the child remote data module.

33.1.7.4.1 TSharedConnection.ConnectEvent Method

Pascal

```
procedure ConnectEvent(Sender: TObject; Connecting: Boolean);
```

C++

```
__fastcall ConnectEvent(TObject * Sender, Boolean Connecting);
```

Description

This is ConnectEvent, a member of class TSharedConnection.

33.1.7.4.2 TSharedConnection.DoConnect Method

Provides the interface for a method that opens a connection.

Pascal

```
procedure DoConnect; override;
```

C++

```
virtual __fastcall DoConnect();
```

Description

The Connected property uses DoConnect to establish a connection. Descendant classes override the DoConnect method to establish their connection as appropriate. As implemented in TCustomConnection, DoConnect does nothing.

See Also

Connected

DoDisconnect

33.1.7.4.3 TSharedConnection.DoDisconnect Method

Provides the interface for a method that terminates the connection.

Pascal

```
procedure DoDisconnect; override;
```

C++

```
virtual __fastcall DoDisconnect();
```

Description

The Connected property uses DoDisconnect to close a connection. Descendant classes override the DoDisconnect method to drop a connection. As implemented in TCustomConnection, DoDisconnect does nothing.

See Also

Connected

DoConnect

33.1.7.4.4 TSharedConnection.GetConnected Method

Returns the value of the Connected property.

Pascal

```
function GetConnected: Boolean; override;
```

C++

```
virtual __fastcall Boolean GetConnected();
```

Description

GetConnected is the protected read implementation of the Connected property. Descendant classes override this method to return true if a connection is open and false otherwise. As defined in TCustomConnection, GetConnected always returns false.

See Also

Connected

SetConnected

33.1.7.4.5 TSharedConnection.GetServer Method

Returns the IAppServer interface of the child remote data module.

Pascal

```
function GetServer: IAppServer; override;
```

C++

```
virtual __fastcall IAppServer GetServer();
```

Description

Call GetServer to obtain the interface of the remote data module on the application server. GetServer returns the IAppServer interface of the same remote data module as the one whose interface is the value of the AppServer property. Client applications can communicate directly with the child remote data module through this interface.

See Also

AppServer

33.1.7.4.6 TSharedConnection.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

33.1.7.5 TSharedConnection Properties

The properties of the TSharedConnection class are listed here.

Published Properties

	Name	Description
	ChildName (see page 1314)	Identifies the child remote data module on the application server with which you want to communicate.
	ParentConnection (see page 1314)	Specifies the connection component that establishes a connection to the main remote data module on the application server.

33.1.7.5.1 TSharedConnection.ChildName Property

Identifies the child remote data module on the application server with which you want to communicate.

Pascal

```
property ChildName: String;
```

C++

```
__property UnicodeString ChildName;
```

Description

Set ChildName to the name of the property on the application server's main remote data module that exposes the interface for the desired child remote data module. TSharedConnection (see page 1309) uses this property to obtain the interface for the child remote data module, which it sets as the value of the AppServer property.

See Also

AppServer

ParentConnection (see page 1314)

33.1.7.5.2 TSharedConnection.ParentConnection Property

Specifies the connection component that establishes a connection to the main remote data module on the application server.

Pascal

```
property ParentConnection: TDispatchConnection;
```

C++

```
__property TDispatchConnection ParentConnection;
```

Description

Set ParentConnection to a connection component that connects to the main remote data module on the application server. TSharedConnection (see page 1309) uses this component's connection for communicating to the child remote data module.

The main remote data module with which ParentConnection communicates must expose the interface for the child remote data module as a property on its IAppServer interface.

See Also

AppServer

ChildName (see page 1314)

33.1.7.6 TSharedConnection Events

The events of the TSharedConnection class are listed here.

Published Events

	Name	Description
	AfterConnect (see page 1315)	Occurs after a connection is established.
	AfterDisconnect (see page 1315)	Occurs after the connection closes.
	BeforeConnect (see page 1315)	Occurs immediately before establishing a connection.
	BeforeDisconnect (see page 1316)	Occurs immediately before the connection closes.

33.1.7.6.1 TSharedConnection.AfterConnect Event

Occurs after a connection is established.

Pascal

```
property AfterConnect;
```

C++

```
__property AfterConnect;
```

Description

Write an AfterConnect event handler to take application-specific actions immediately after the connection component opens a connection to the remote source of database information.

See Also

BeforeConnect

Connected

Connecting to a Database Server

33.1.7.6.2 TSharedConnection.AfterDisconnect Event

Occurs after the connection closes.

Pascal

```
property AfterDisconnect;
```

C++

```
__property AfterDisconnect;
```

Description

Write an AfterDisconnect event handler to take application-specific actions after the connection component drops a connection.

See Also

BeforeDisconnect

Connected

Disconnecting from a Database Server

33.1.7.6.3 TSharedConnection.BeforeConnect Event

Occurs immediately before establishing a connection.

Pascal

```
property BeforeConnect;
```

C++

```
__property BeforeConnect;
```

Description

Write a BeforeConnect event handler to take application-specific actions before the connection component opens a connection to the remote source of database information.

See Also

AfterConnect

Connected

Connecting to a Database Server

33.1.7.6.4 TSharedConnection.BeforeDisconnect Event

Occurs immediately before the connection closes.

Pascal

```
property BeforeDisconnect;
```

C++

```
__property BeforeDisconnect;
```

Description

Write a BeforeDisconnect event handler to take application-specific actions before dropping a connection.

Delphi Examples:

```
{
This BeforeDisconnect event handler makes sure that the
client dataset has applied all its pending updates before
terminating the connection to the server.
}
procedure TForm1.RemoteServer1BeforeDisconnect(Sender: TObject);
begin
  if (ClientDataSet1.ChangeCount > 0) then
    ClientDataSet1.ApplyUpdates(-1);
end;
```

See Also

AfterDisconnect

Connected

Disconnecting from a Database Server

33.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	GetMIDASAppServerList (see page 1317)	This is function MConnect.GetMIDASAppServerList.

33.2.1 MConnect.GetMIDASAppServerList Function

Pascal

```
procedure GetMIDASAppServerList(List: TStringList; const RegCheck: string);
```

C++

```
GetMIDASAppServerList(TStringList List, const UnicodeString RegCheck);
```

File

MConnect

Description

This is function MConnect.GetMIDASAppServerList.

34 Menus Namespace

This is namespace Menus.

Topics

Name	Description
Classes (see page 1321)	The following table lists classes in this documentation.
Functions (see page 1446)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 1460)	The following table lists structs, records, enums in this documentation.
Types (see page 1465)	The following table lists types in this documentation.
Variables (see page 1470)	The following table lists variables in this documentation.
Constants (see page 1472)	The following table lists constants in this documentation.

Classes

	Name	Description
	EMenuError (see page 1321)	EMenuError is the exception class for errors involving menu items.
	TMainMenu (see page 1322)	TMainMenu encapsulates a menu bar and its accompanying drop-down menus for a form.
	TMenu (see page 1331)	TMenu is the base type for menu components such as TMainMenu (see page 1322) and TPopupMenu (see page 1430).
	TMenuActionLink (see page 1352)	TMenuActionLink links menu item clients to the corresponding menu action.
	TMenuItem (see page 1363)	TMenuItem describes the properties of an item in a menu.
	TMenuItemEnumerator (see page 1424)	This is class Menus.TMenuItemEnumerator.
	TMenuItemStack (see page 1426)	This is class Menus.TMenuItemStack.
	TPopupList (see page 1427)	TPopupList manages the window that handles popup menu messages.
	TPopupMenu (see page 1430)	TPopupMenu encapsulates the properties, methods, and events of a pop-up menu.

Constants

Name	Description
cDialogSuffix (see page 1472)	This is constant Menus.cDialogSuffix.
cHotkeyPrefix (see page 1472)	This is constant Menus.cHotkeyPrefix.
cLineCaption (see page 1472)	This is constant Menus.cLineCaption.

Functions

	Name	Description
	AnsiSameCaption (see page 1446)	Indicates whether two strings are the same based on the current locale, ignoring case information and accelerator characters.

	DrawMenuItem (see page 1447)	This is function Menus.DrawMenuItem.
	GetHotkey (see page 1447)	Returns the last character in a string marked with an ampersand (&).
	IsAltGRPressed (see page 1447)	This is function Menus.IsAltGRPressed.
	NewItem (see page 1448)	Creates and initializes a menu item.
	NewLine (see page 1449)	Creates a menu item for a separator.
	NewMenu (see page 1450)	Creates and initializes a main menu.
	NewPopupMenu (see page 1451)	Creates and initializes a popup menu.
	NewSubMenu (see page 1452)	Creates and initializes a menu item that represents a submenu.
	SameCaption (see page 1453)	This is function Menus.SameCaption.
	ShortCut (see page 1453)	Creates a menu shortcut for a specified key combination.
	ShortCutFromMessage (see page 1454)	This is function Menus.ShortCutFromMessage.
	ShortCutToKey (see page 1454)	Returns the key code and shift state of a menu shortcut.
	ShortCutToText (see page 1456)	Converts a menu shortcut into a string that describes the shortcut.
	StripHotkey (see page 1457)	Removes the accelerator character marker from a string.
	TextToShortCut (see page 1457)	Creates a menu shortcut from a text string.

Structs, Records, Enums

	Name	Description
	TFindItemKind (see page 1460)	TFindItemKind indicates the criterion to use when locating menu items.
	TMenuAnimations (see page 1461)	TMenuAnimation (see page 1466) and TMenuAnimations indicate how a popup menu appears on screen.
	TMenuBreak (see page 1461)	TMenuBreak indicates the criterion to use when locating menu items.
	TMenuItemAutoFlag (see page 1462)	TMenuItemAutoFlag and TMenuAutoFlag (see page 1467) Indicate when menu items are automatically maintained.
	TPopupAlignment (see page 1463)	TPopupAlignment represents the position of a popup menu relative to the mouse.
	TTrackButton (see page 1463)	TTrackButton specifies which mouse button activates a tool button menu.

Types

Name	Description
PMenuItem (see page 1465)	This is type Menus.PMenuItem.
TAdvancedMenuDrawItemEvent (see page 1465)	TAdvancedMenuDrawItemEvent is the type for event handlers that render menu items in owner-drawn menus.
TMenuActionLinkClass (see page 1466)	This is type Menus.TMenuActionLinkClass.
TMenuAnimation (see page 1466)	TMenuAnimation and TMenuAnimations (see page 1461) indicate how a popup menu appears on screen.
TMenuAutoFlag (see page 1467)	TMenuItemAutoFlag (see page 1462) and TMenuAutoFlag Indicate when menu items are automatically maintained.
TMenuChangeEvent (see page 1467)	This is type Menus.TMenuChangeEvent.
TMenuDrawItemEvent (see page 1468)	TMenuDrawItemEvent is the type for event handlers that render menu items in owner-drawn menus.
TMenuHandle (see page 1468)	This is type Menus.TMenuHandle.
TMenuMeasureItemEvent (see page 1469)	TMenuMeasureItemEvent is the type for event handlers that render menu items in owner-Measuren menus.

Variables

Name	Description
PopupList (see page 1470)	Provides centralized handling of Windows messages directed at popup menus.
ShortCutItems (see page 1470)	This is variable Menus.ShortCutItems.
ValidMenuHotkeys (see page 1471)	Determines what accelerator keys can be automatically assigned to a menu item caption.

34.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	EMenuError (see page 1321)	EMenuError is the exception class for errors involving menu items.
	TMainMenu (see page 1322)	TMainMenu encapsulates a menu bar and its accompanying drop-down menus for a form.
	TMenu (see page 1331)	TMenu is the base type for menu components such as TMainMenu (see page 1322) and TPopupMenu (see page 1430).
	TMenuActionLink (see page 1352)	TMenuActionLink links menu item clients to the corresponding menu action.
	TMenuItem (see page 1363)	TMenuItem describes the properties of an item in a menu.
	TMenuItemEnumerator (see page 1424)	This is class Menus.TMenuItemEnumerator.
	TMenuItemStack (see page 1426)	This is class Menus.TMenuItemStack.
	TPopupList (see page 1427)	TPopupList manages the window that handles popup menu messages.
	TPopupMenu (see page 1430)	TPopupMenu encapsulates the properties, methods, and events of a pop-up menu.

34.1.1 EMenuError Class

EMenuError is the exception class for errors involving menu items.

Class Hierarchy

Pascal

```
EMenuError = class(Exception);
```

C++

```
class EMenuError : public Exception;
```

File

Menus

Description

EMenuError is raised if an error occurs while an application is working with menu items. For example, an EMenuError exception occurs if

An application tries to delete an item from an empty menu

A menu item attempts to insert itself into it's own list of items

An application attempts to reference a menu that does not exist

An application specifies an invalid GroupIndex for a menu item

See Also

TMenuItem (see page 1363)

TMenu (see page 1331)

34.1.2 TMainMenu Class

TMainMenu encapsulates a menu bar and its accompanying drop-down menus for a form.

Class Hierarchy

Pascal

```
TMainMenu = class(TMenu);
```

C++

```
class TMainMenu : public TMenu;
```

File

Menus

Description

Use TMainMenu to provide the main menu for a form. To begin designing a menu, add a main menu to a form, and double-click the component.

TMainMenu introduces properties and methods for merging the drop-down menus of the main menu with the main menu of another form.

TMainMenu introduces properties and methods for assisting in the menu negotiation process of an OLE container.

Tip: To create a menu that users can configure at runtime, use the TActionMainMenuBar component with an action manager instead.

See Also

TMenuItem (see page 1363)

TPopupMenu (see page 1430)

TActionMainMenuBar

TActionManager

34.1.2.1 TMainMenu Members

The following tables list the members exposed by TMainMenu.

Protected Methods

	Name	Description
	AdjustBiDiBehavior (see page 1335)	Applies the BiDiMode (see page 1344) property to the menu when the Windows control is created.
	DoBiDiModeChanged (see page 1336)	Updates the menu when the BiDiMode (see page 1344) property changes.
	DoChange (see page 1336)	Generates an OnChange (see page 1352) event.
	DoGetMenuString (see page 1337)	Returns the caption for an item in the menu.
	GetHandle (see page 1324)	Returns the value of the Handle property.
	IsOwnerDraw (see page 1339)	Indicates whether menu items can contain more than simple captions.
	Loaded (see page 1340)	Calls the DoChange (see page 1336) method when the menu is first loaded into memory.
	MenuChanged (see page 1325)	Responds when the menu changes.

	Notification (see page 1341)	Responds when components are added or deleted.
	SetChildOrder (see page 1342)	Ensures that the menu item indexes reflect the order in which they are loaded and saved.
	UpdateItems (see page 1343)	Causes all menu items to update themselves.

Public Methods

	Name	Description
	Create (see page 1333)	Creates an instance of TMenu (see page 1331).
	CreateMenuItem (see page 1335)	This is CreateMenuItem, a member of class TMenu.
	Destroy (see page 1334)	Destroys the menu component.
	DispatchCommand (see page 1335)	Calls the Click method of the menu item with the specified value as its Command property.
	DispatchPopup (see page 1336)	Dispatches a pop-up menu command.
	FindItem (see page 1337)	Locates and returns a menu item.
	GetChildren (see page 1338)	Executes a callback for child menu items.
	GetHelpContext (see page 1338)	Returns the help context ID associated with a menu item.
	GetOle2AcceleratorTable (see page 1324)	Returns the main menu's accelerator table.
	IsRightToLeft (see page 1339)	Indicates whether menu items read from right to left.
	IsShortCut (see page 1339)	Dispatches a key message if it represents a shortcut key for an item in the menu.
	Merge (see page 1325)	Combines the main menu of one form with the main menu of another in non-MDI applications.
	ParentBiDiModeChanged (see page 1341)	This is the overview for the ParentBiDiModeChanged method overload.
	PopulateOle2Menu (see page 1326)	Populates an OLE menu with items from the main menu.
	ProcessMenuChar (see page 1342)	Responds to keyboard input for the menu.
	SetOle2MenuHandle (see page 1326)	Associates the handle for an OLE menu with the main menu.
	Unmerge (see page 1326)	Reverses the merging of two menus in a non-MDI application.

Public Properties

	Name	Description
	Handle (see page 1345)	Provides access to the Windows menu handle for the menu.
	WindowHandle (see page 1351)	Provides access to the handle of the window that uses the menu.

Published Events

	Name	Description
	OnChange (see page 1331)	Occurs when the menu changes.

Published Properties

	Name	Description
	AutoHotkeys (see page 1327)	Determines whether the accelerator keys for menu items can be reset automatically.
	AutoLineReduction (see page 1328)	Determines whether redundant separator bars are automatically removed from the menu.
	AutoMerge (see page 1328)	Determines if the main menus of secondary forms merge with the main menu of the main form in non-MDI applications at runtime.
	BiDiMode (see page 1329)	Determines the layout of the menu in Middle East locales.
	Images (see page 1329)	Lists the images that can appear beside individual menu items.
	Items (see page 1346)	Describes the elements of the menu.
	OwnerDraw (see page 1330)	Specifies whether the menu items are drawn by the application.
	ParentBiDiMode (see page 1330)	Indicates whether the BiDiMode property is inherited from the control that uses the menu.

34.1.2.2 TMainMenu Methods

The methods of the TMainMenu class are listed here.

Protected Methods

	Name	Description
	GetHandle (see page 1324)	Returns the value of the Handle property.
	MenuChanged (see page 1325)	Responds when the menu changes.

Public Methods

	Name	Description
	GetOle2AcceleratorTable (see page 1324)	Returns the main menu's accelerator table.
	Merge (see page 1325)	Combines the main menu of one form with the main menu of another in non-MDI applications.
	PopulateOle2Menu (see page 1326)	Populates an OLE menu with items from the main menu.
	SetOle2MenuHandle (see page 1326)	Associates the handle for an OLE menu with the main menu.
	Unmerge (see page 1326)	Reverses the merging of two menus in a non-MDI application.

34.1.2.2.1 TMainMenu.GetHandle Method

Returns the value of the Handle property.

Pascal

```
function GetHandle: HMENU; override;
```

C++

```
virtual __fastcall HMENU GetHandle();
```

Description

GetHandle is the protected implementation of the Handle property. Override this method to change the implementation of the handle property. For example, TMainMenu overrides GetHandle to adjust for menus added by embedded OLE objects.

See Also

Handle (see page 1345)

34.1.2.2.2 TMainMenu.GetOle2AcceleratorTable Method

Returns the main menu's accelerator table.

Pascal

```
procedure GetOle2AcceleratorTable(var AccelTable: HAccel; var AccelCount: Integer; const
Groups: array of Integer);
```

C++

```
__fastcall GetOle2AcceleratorTable(HAccel AccelTable, int AccelCount, const array of Integer
Groups);
```

Description

Call GetOle2AcceleratorTable to obtain a handle to an accelerator table for all menu items with a GroupIndex value that appears in the Groups parameter. The AccelTable parameter is set to the accelerator table handle, and the AccelCount is set to the number of entries in the table. GetOle2AcceleratorTable is used by OLE containers when merging menus.

See Also

GroupIndex (see page 1408)

ShortCut (see page 1415)

34.1.2.2.3 TMainMenu.MenuChanged Method

Responds when the menu changes.

Pascal

```
procedure MenuChanged(Sender: TObject; Source: TMenuItem; Rebuild: Boolean); override;
```

C++

```
virtual __fastcall MenuChanged(TObject * Sender, TMenuItem Source, Boolean Rebuild);
```

Description

MenuChanged is called automatically when changes occur that affect the structure of the menu. The Sender parameter indicates the object that originates the change. It's value can be nil (Delphi) or NULL (C++). Source indicates which menu item is affected. Rebuild indicates whether the menu must be rebuilt to reflect the change (for example when items are added or deleted).

MenuChanged calls the DoChange method, which generates an OnChange event.

Note: Changes that affect individual items rather than the menu as a whole trigger the UpdateItems method instead of MenuChanged. For example, when the Images property changes, UpdateItems is called rather than MenuChanged.

See Also

Items (see page 1346)

UpdateItems (see page 1343)

DoChange (see page 1336)

34.1.2.2.4 TMainMenu.Merge Method

Combines the main menu of one form with the main menu of another in non-MDI applications.

Pascal

```
procedure Merge(Menu: TMainMenu);
```

C++

```
__fastcall Merge(TMainMenu Menu);
```

Description

Call Merge to merge another main menu with this one. Specify the menu you want merged with this menu as the Menu parameter.

For example, to use the main menu of the first form in an application as the main menu for the application, call Merge to merge the main menu of any additional forms in the application with the main menu of the first form.

Depending on the value of the GroupIndex property of menu items on the main menu, the merged menu items can replace menu items on the menu bar, or insert menu items into the menu bar. See GroupIndex for information on how to do these things.

To merge and unmerge main menus automatically when other forms are displayed, change the value of the AutoMerge (see page 1328) property to true on all forms whose menus should be merged in.

See Also

AutoMerge (see page 1328)

GroupIndex (see page 1408)

Unmerge (see page 1326)

34.1.2.2.5 TMainMenu.PopulateOle2Menu Method

Populates an OLE menu with items from the main menu.

Pascal

```
procedure PopulateOle2Menu(SharedMenu: HMenu; const Groups: array of Integer; var Widths:
array of Longint);
```

C++

```
__fastcall PopulateOle2Menu(HMenu SharedMenu, const array of Integer Groups, array of Longint
Widths);
```

Description

Call PopulateOle2Menu to merge the menu items with GroupIndex values specified by the Groups parameter into the menu that uses the SharedMenu handle. PopulateOle2Menu appends all menu items in the specified groups to the menu, and increments the corresponding entries in the Widths array for every menu item appended. PopulateOle2Menu is used by OLE containers when merging menus.

Note: PopulateOle2Menu populates the menu specified by the SharedMenu parameter but does not set this as the OLE menu associated with the menu.

See Also

GetOle2AcceleratorTable (see page 1324)

SetOle2MenuHandle (see page 1326)

GroupIndex (see page 1408)

34.1.2.2.6 TMainMenu.SetOle2MenuHandle Method

Associates the handle for an OLE menu with the main menu.

Pascal

```
procedure SetOle2MenuHandle(Handle: HMENU);
```

C++

```
__fastcall SetOle2MenuHandle(HMENU Handle);
```

Description

Call SetOle2MenuHandle to temporarily replace the menu of a form with the menu represented by the Handle (see page 1345) parameter. To restore the original menu, call SetOle2MenuHandle with the Handle (see page 1345) parameter set to 0. SetOle2MenuHandle allows an OLE container to use menu items provided by an OLE server.

See Also

GetOle2AcceleratorTable (see page 1324)

PopulateOle2Menu (see page 1326)

34.1.2.2.7 TMainMenu.Unmerge Method

Reverses the merging of two menus in a non-MDI application.

Pascal

```
procedure Unmerge(Menu: TMainMenu);
```

C++

```
__fastcall Unmerge(TMainMenu Menu);
```

Description

Call Unmerge to remove items from another main menu that were added using the Merge (see page 1325) method. The Menu parameter is the merged menu whose items should be removed from the main menu.

See Also

AutoMerge (see page 1328)

Merge (see page 1325)

34.1.2.3 TMainMenu Properties

The properties of the TMainMenu class are listed here.

Published Properties

	Name	Description
	AutoHotkeys (see page 1327)	Determines whether the accelerator keys for menu items can be reset automatically.
	AutoLineReduction (see page 1328)	Determines whether redundant separator bars are automatically removed from the menu.
	AutoMerge (see page 1328)	Determines if the main menus of secondary forms merge with the main menu of the main form in non-MDI applications at runtime.
	BiDiMode (see page 1329)	Determines the layout of the menu in Middle East locales.
	Images (see page 1329)	Lists the images that can appear beside individual menu items.
	OwnerDraw (see page 1330)	Specifies whether the menu items are drawn by the application.
	ParentBiDiMode (see page 1330)	Indicates whether the BiDiMode property is inherited from the control that uses the menu.

34.1.2.3.1 TMainMenu.AutoHotkeys Property

Determines whether the accelerator keys for menu items can be reset automatically.

Pascal

```
property AutoHotkeys: TMenuAutoFlag;
```

C++

```
__property TMenuAutoFlag AutoHotkeys;
```

Description

Set AutoHotkeys to indicate whether the menu should automatically ensure that accelerator keys for its items are reset if necessary so that every menu item has an accelerator key and no two items have the same accelerator key. This ensures that menu items added dynamically at runtime can have accelerator keys that do not introduce conflicts. TMenu tries to preserve all existing accelerator key mappings when adjusting the accelerator keys, but may be forced to change some in order to ensure that every item has a unique accelerator.

Before the menu is displayed, the menu checks and adjusts the accelerator key mappings if AutoHotkeys is maAutomatic. When AutoHotkeys is maManual, the menu does not alter the top-level menu items, but may adjust the accelerator keys for submenus, depending on the AutoHotkeys property of the menu item that is the parent of the submenu.

When AutoHotkeys is maManual, an application can still adjust the menu items by calling the Items property's RethinkHotkeys method. AutoHotkeys merely controls whether this adjustment occurs automatically.

Note: The AutoHotkeys property of a menu is linked to the AutoHotkeys property of the TMenuItems object that implements its

Items property. Setting one sets the other.

See Also

AutoHotkeys (see page 1394)

Items (see page 1346)

RethinkHotkeys (see page 1388)

34.1.2.3.2 TMainMenu.AutoLineReduction Property

Determines whether redundant separator bars are automatically removed from the menu.

Pascal

```
property AutoLineReduction: TMenuAutoFlag;
```

C++

```
__property TMenuAutoFlag AutoLineReduction;
```

Description

Set AutoLineReduction to indicate whether the menu should automatically ensure that the menu does not start or end with a separator bar and that two separator bars do not appear next to each other. This ensures that menus built dynamically at runtime do not have separator bars in undesired positions.

Before the menu is displayed, it removes superfluous separator bars when AutoLineReduction is maAutomatic. When AutoLineReduction is maManual, the menu does not remove separators from the top-level menu items, but may remove superfluous separators from submenus, depending on the AutoLineReduction property of the menu item that is the parent of the submenu.

When AutoLineReduction is maManual, an application can still remove superfluous separators by calling the Items property's RethinkLines method. AutoLineReduction merely determines whether this adjustment occurs automatically.

Note: The AutoLineReduction property of a menu is linked to the AutoLineReduction property of the TMenuItems object that implements its Items property. Setting one sets the other.

See Also

AutoLineReduction (see page 1395)

Items (see page 1346)

RethinkLines (see page 1389)

34.1.2.3.3 TMainMenu.AutoMerge Property

Determines if the main menus of secondary forms merge with the main menu of the main form in non-MDI applications at runtime.

Pascal

```
property AutoMerge: Boolean;
```

C++

```
__property Boolean AutoMerge;
```

Description

To merge a form's menus with the main menu in the main form, set the AutoMerge property of each main menu you want merged to true. Make sure that the AutoMerge property of the main menu of the main form (the one other menus are merged

into) remains false. How menus merge depends on the value of the GroupIndex property for each menu item.

If the application is an MDI application (the FormStyle properties are set so the main form is a parent form and subsequent forms are child forms), menu merging occurs automatically without using the AutoMerge property.

See Also

FormStyle

GroupIndex (see page 1408)

Merge (see page 1325)

34.1.2.3.4 TMainMenu.BiDiMode Property

Determines the layout of the menu in Middle East locales.

Pascal

```
property BiDiMode: TBiDiMode;
```

C++

```
__property TBiDiMode BiDiMode;
```

Description

Set BiDiMode to control the layout of menu items. BiDiMode controls whether the menu items read from left to right or right to left.

See Also

ParentBiDiMode (see page 1351)

IsRightToLeft (see page 1339)

34.1.2.3.5 TMainMenu.Images Property

Lists the images that can appear beside individual menu items.

Pascal

```
property Images: TCustomImageList;
```

C++

```
__property TCustomImageList Images;
```

Description

Use Images to hold the set of images that can appear beside menu items in the menu. Individual menu items set their ImageIndex property to indicate which image in the list appears to the left of the menu item caption.

Images in the list should be 16x16.

Tip: If the application uses action lists, this property should be the value of the action list's Images property. This way, menu items and buttons that do the same thing can display the same image.

See Also

ImageIndex (see page 1411)

Images

Items (see page 1346)

34.1.2.3.6 TMainMenu.OwnerDraw Property

Specifies whether the menu items are drawn by the application.

Pascal

```
property OwnerDraw: Boolean;
```

C++

```
__property Boolean OwnerDraw;
```

Description

Set OwnerDraw to true to indicate that the application will draw menu items in an OnDrawItem event handler. When OwnerDraw is true, menu items receive an OnMeasureItem and an OnDrawItem event when they need to be rendered on screen.

Set OwnerDraw to false to accept the default drawing. When OwnerDraw is false, menu items display the caption with an optional image to the left (If the Images and ImageIndex properties are set).

Owner-draw menus are useful for displaying the value that appears when selecting a menu item. For example, in a menu that allows users to choose a color, an owner-draw menu could display rectangles drawn in the designated color, rather than color names.

See Also

OnMeasureItem (see page 1423)

OnDrawItem (see page 1422)

Images (see page 1345)

34.1.2.3.7 TMainMenu.ParentBiDiMode Property

Indicates whether the BiDiMode property is inherited from the control that uses the menu.

Pascal

```
property ParentBiDiMode: Boolean;
```

C++

```
__property Boolean ParentBiDiMode;
```

Description

By default, ParentBiDiMode is true, indicating that the layout of menu items follows the BiDiMode setting of the associated form or control. Change ParentBiDiMode to false to override the default BiDiMode setting and control the menu's layout independently.

See Also

BiDiMode (see page 1344)

34.1.2.4 TMainMenu Events

The events of the TMainMenu class are listed here.

Published Events

	Name	Description
	OnChange (see page 1331)	Occurs when the menu changes.

34.1.2.4.1 TMainMenu.OnChange Event

Occurs when the menu changes.

Pascal

```
property OnChange: TMenuChangeEvent;
```

C++

```
__property TMenuChangeEvent OnChange;
```

Description

Write an OnChange event handler to respond to changes in the menu's structure. OnChange occurs when the menu is loaded into memory, and in response to property changes that affect the menu's structure.

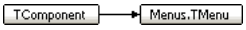
See Also

Items (see page 1346)

34.1.3 TMenu Class

TMenu is the base type for menu components such as TMainMenu (see page 1322) and TPopupMenu (see page 1430).

Class Hierarchy



Pascal

```
[RootDesignerSerializerAttribute('', '', False)]
TMenu = class(TComponent);
```

C++

```
[RootDesignerSerializerAttribute('', '', False)]
class TMenu : public TComponent;
```

File

Menus

Description

Use TMenu as a base class when defining a component that represents a collection of menu items. TMenu introduces properties and methods to create a Windows menu that represents a set of menu items, dispatch commands associated with those menu items, and locate the help context id for any of those menu items.

See Also

TMainMenu (see page 1322)

TMenuItem (see page 1363)

TPopupMenu (see page 1430)

Disabling Menu Items

Adding Images to Menu Items

Adding

Building Menus

Determining the Order of Merged Menu Items: GroupIndex Property
Editing Menu Items in the Object Inspector
Importing Resource Files
Manipulating Menu Items at Runtime
Merging Menus
Moving Menu Items
Naming Conventions for Template Menu Items and Event Handlers
Naming Menus
Naming the Menu Items
Opening the Menu Designer
Saving a Menu as a Template
Specifying Accelerator Keys and Keyboard Shortcuts
Specifying the Active Menu: Menu Property
Switching Between Menus at Design Time
Using Menu Templates
Using the Menu Designer Context Menu
Viewing the Menu

34.1.3.1 TMenu Members

The following tables list the members exposed by TMenu.

Protected Events

	Name	Description
	OnChange (see page 1352)	Occurs when the menu changes.

Protected Methods

	Name	Description
	AdjustBiDiBehavior (see page 1335)	Applies the BiDiMode (see page 1344) property to the menu when the Windows control is created.
	DoBiDiModeChanged (see page 1336)	Updates the menu when the BiDiMode (see page 1344) property changes.
	DoChange (see page 1336)	Generates an OnChange (see page 1352) event.
	DoGetMenuString (see page 1337)	Returns the caption for an item in the menu.
	GetHandle (see page 1338)	Returns the value of the Handle (see page 1345) property.
	IsOwnerDraw (see page 1339)	Indicates whether menu items can contain more than simple captions.
	Loaded (see page 1340)	Calls the DoChange (see page 1336) method when the menu is first loaded into memory.
	MenuChanged (see page 1340)	Responds when the menu changes.
	Notification (see page 1341)	Responds when components are added or deleted.
	SetChildOrder (see page 1342)	Ensures that the menu item indexes reflect the order in which they are loaded and saved.
	UpdateItems (see page 1343)	Causes all menu items to update themselves.

Public Methods

	Name	Description
	Create (see page 1333)	Creates an instance of TMenu (see page 1331).
	CreateMenuItem (see page 1335)	This is CreateMenuItem, a member of class TMenu.
	Destroy (see page 1334)	Destroys the menu component.
	DispatchCommand (see page 1335)	Calls the Click method of the menu item with the specified value as its Command property.
	DispatchPopup (see page 1336)	Dispatches a pop-up menu command.
	FindItem (see page 1337)	Locates and returns a menu item.
	GetChildren (see page 1338)	Executes a callback for child menu items.
	GetHelpContext (see page 1338)	Returns the help context ID associated with a menu item.
	IsRightToLeft (see page 1339)	Indicates whether menu items read from right to left.
	IsShortCut (see page 1339)	Dispatches a key message if it represents a shortcut key for an item in the menu.
	ParentBiDiModeChanged (see page 1341)	This is the overview for the ParentBiDiModeChanged method overload.
	ProcessMenuChar (see page 1342)	Responds to keyboard input for the menu.

Public Properties

	Name	Description
	AutoHotkeys (see page 1343)	Determines whether the accelerator keys for menu items can be reset automatically.
	AutoLineReduction (see page 1344)	Determines whether redundant separator bars are automatically removed from the menu.
	BiDiMode (see page 1344)	Determines the layout of the menu in Middle East locales.
	Handle (see page 1345)	Provides access to the Windows menu handle for the menu.
	Images (see page 1345)	Lists the images that can appear beside individual menu items.
	OwnerDraw (see page 1350)	Specifies whether the menu items are drawn by the application.
	ParentBiDiMode (see page 1351)	Indicates whether the BiDiMode (see page 1344) property is inherited from the control that uses the menu.
	WindowHandle (see page 1351)	Provides access to the handle of the window that uses the menu.

Published Properties

	Name	Description
	Items (see page 1346)	Describes the elements of the menu.

34.1.3.2 TMenu.Create Constructor

Creates an instance of TMenu (see page 1331).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TMenu(TComponent * AOwner);
```

Description

Call Create to create a menu at runtime. Menus (see page 1319) placed on forms at design time are created automatically. Create calls the inherited Create method, then sets the initial values for the menu component, including creating an Items (see page 1346) property.

The AOwner parameter indicates the component (usually a form) that is responsible for managing the memory associated with the menu.

See Also

Items (see page 1346)

34.1.3.3 TMenu.Destroy Destructor

Destroys the menu component.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TMenu();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TMenu (see page 1331) reference is not nil, and only then calls Destroy. Destroy frees the Items (see page 1346) property, and then calls the inherited Destroy method.

See Also

System::TObject::Free

Items (see page 1346)

34.1.3.4 TMenu Methods

The methods of the TMenu class are listed here.

Protected Methods

	Name	Description
	AdjustBiDiBehavior (see page 1335)	Applies the BiDiMode (see page 1344) property to the menu when the Windows control is created.
	DoBiDiModeChanged (see page 1336)	Updates the menu when the BiDiMode (see page 1344) property changes.
	DoChange (see page 1336)	Generates an OnChange (see page 1352) event.
	DoGetMenuString (see page 1337)	Returns the caption for an item in the menu.
	GetHandle (see page 1338)	Returns the value of the Handle (see page 1345) property.
	IsOwnerDraw (see page 1339)	Indicates whether menu items can contain more than simple captions.
	Loaded (see page 1340)	Calls the DoChange (see page 1336) method when the menu is first loaded into memory.
	MenuChanged (see page 1340)	Responds when the menu changes.
	Notification (see page 1341)	Responds when components are added or deleted.
	SetChildOrder (see page 1342)	Ensures that the menu item indexes reflect the order in which they are loaded and saved.
	UpdateItems (see page 1343)	Causes all menu items to update themselves.

Public Methods

	Name	Description
	CreateMenuItem (see page 1335)	This is CreateMenuItem, a member of class TMenu.
	DispatchCommand (see page 1335)	Calls the Click method of the menu item with the specified value as its Command property.
	DispatchPopup (see page 1336)	Dispatches a pop-up menu command.
	FindItem (see page 1337)	Locates and returns a menu item.
	GetChildren (see page 1338)	Executes a callback for child menu items.
	GetHelpContext (see page 1338)	Returns the help context ID associated with a menu item.
	IsRightToLeft (see page 1339)	Indicates whether menu items read from right to left.
	IsShortCut (see page 1339)	Dispatches a key message if it represents a shortcut key for an item in the menu.
	ParentBiDiModeChanged (see page 1341)	This is the overview for the ParentBiDiModeChanged method overload.
	ProcessMenuChar (see page 1342)	Responds to keyboard input for the menu.

34.1.3.4.1 TMenu.AdjustBiDiBehavior Method

Applies the BiDiMode (see page 1344) property to the menu when the Windows control is created.

Pascal

```
procedure AdjustBiDiBehavior;
```

C++

```
__fastcall AdjustBiDiBehavior();
```

Description

AdjustBiDiBehavior is called automatically when the Windows control for the menu is created. It ensures that the BiDiMode (see page 1344) property is properly reflected in the menu if the application is running in a Middle Eastern locale.

See Also

BiDiMode (see page 1344)

34.1.3.4.2 TMenu.CreateMenuItem Method

Pascal

```
function CreateMenuItem: TMenuItem; dynamic;
```

C++

```
__fastcall TMenuItem CreateMenuItem();
```

Description

This is CreateMenuItem, a member of class TMenu.

34.1.3.4.3 TMenu.DispatchCommand Method

Calls the Click method of the menu item with the specified value as its Command property.

Pascal

```
function DispatchCommand(ACommand: Word): Boolean;
```

C++

```
__fastcall Boolean DispatchCommand(Word ACommand);
```

Description

Call DispatchCommand to simulate a click on the menu item associated with a command ID. If DispatchCommand locates a menu item that matches the command ID, it calls the Click method for that menu item and returns true. If there is no such item, DispatchCommand returns false.

See Also

DispatchPopup (see page 1336)

IsShortCut (see page 1339)

Items (see page 1346)

Click (see page 1375)

Command (see page 1401)

34.1.3.4.4 TMenu.DispatchPopup Method

Dispatches a pop-up menu command.

Pascal

```
function DispatchPopup(AHandle: HMENU): Boolean;
```

C++

```
__fastcall Boolean DispatchPopup(HMENU AHandle);
```

Description

Call DispatchPopup to simulate a click on the menu item associated with the pop-up menu handle. If DispatchPopup locates a menu item that matches the pop-up menu handle, it calls the Click method for that menu item and returns true. If there is no such item, DispatchPopup returns false.

See Also

DispatchCommand (see page 1335)

IsShortCut (see page 1339)

Items (see page 1346)

Click (see page 1375)

34.1.3.4.5 TMenu.DoBiDiModeChanged Method

Updates the menu when the BiDiMode (see page 1344) property changes.

Pascal

```
procedure DoBiDiModeChanged;
```

C++

```
__fastcall DoBiDiModeChanged();
```

Description

Do not call DoBiDiModeChanged. It is called automatically when the menu's BiDiMode (see page 1344) property changes to update the menu to reflect the current setting.

See Also

BiDiMode (see page 1344)

34.1.3.4.6 TMenu.DoChange Method

Generates an OnChange (see page 1352) event.

Pascal

```
procedure DoChange(Source: TMenuItem; Rebuild: Boolean); virtual;
```

C++

```
virtual __fastcall DoChange(TMenuItem Source, Boolean Rebuild);
```

Description

DoChange is called automatically when the properties of any menu items change. The Source parameter specifies the menu item (if any) that changed. The Rebuild parameter indicates whether the change requires the menu to regenerate its items.

See Also

OnChange (see page 1352)

MenuChanged (see page 1340)

34.1.3.4.7 TMenu.DoGetMenuString Method

Returns the caption for an item in the menu.

Pascal

```
function DoGetMenuString(Menu: HMENU; ItemID: UINT; Str: PChar; MaxCount: Integer; Flag:
UINT): Integer;
```

C++

```
__fastcall int DoGetMenuString(HMENU Menu, UINT ItemID, const char * Str, int MaxCount, UINT
Flag);
```

Description

DoGetMenuString is used internally to obtain menu item captions, given the Windows menu handle. The Menu parameter indicates the menu's handle. The ItemID parameter specifies the menu item whose caption is desired. If the Flag parameter is MF_BYCOMMAND, ItemID is the Command property of the menu item. If Flag is MF_BYPOSITION, ItemID is the 0-based index of the menu item. The caption is returned in a buffer pointed to by the Str parameter. MaxCount indicates the size of that buffer.

Unlike the Windows API function GetMenuString, DoGetMenuString returns the correct caption for owner-draw menu items.

See Also

Caption (see page 1396)

Command (see page 1401)

34.1.3.4.8 TMenu.FindItem Method

Locates and returns a menu item.

Pascal

```
function FindItem(Value: Integer; Kind: TFindItemKind): TMenuItem;
```

C++

```
__fastcall TMenuItem FindItem(int Value, TFindItemKind Kind);
```

Description

Call FindItem to locate a menu item in the menu or one of its submenus. The menu item is identified by either a pop-up menu handle, command ID, or menu shortcut value. The Kind parameter specifies how the menu item is to be identified.

See Also

TMenuItem (see page 1363)

Command (see page 1401)

Handle (see page 1409)

ShortCut (see page 1415)

34.1.3.4.9 TMenu.GetChildren Method

Executes a callback for child menu items.

Pascal

```
procedure GetChildren(Proc: TGetChildProc; Root: TComponent); override;
```

C++

```
virtual __fastcall GetChildren(TGetChildProc Proc, TComponent * Root);
```

Description

GetChildren is called automatically by the streaming system so that child menu items are saved and read with their parent menu.

See Also

GetParentComponent

SetChildOrder (see page 1342)

34.1.3.4.10 TMenu.GetHandle Method

Returns the value of the Handle (see page 1345) property.

Pascal

```
function GetHandle: HMENU; virtual;
```

C++

```
virtual __fastcall HMENU GetHandle();
```

Description

GetHandle is the protected implementation of the Handle (see page 1345) property. Override this method to change the implementation of the handle property. For example, TMainMenu (see page 1322) overrides GetHandle to adjust for menus added by embedded OLE objects.

See Also

Handle (see page 1345)

34.1.3.4.11 TMenu.GetHelpContext Method

Returns the help context ID associated with a menu item.

Pascal

```
function GetHelpContext(Value: Integer; ByCommand: Boolean): THelpContext;
```

C++

```
__fastcall THelpContext GetHelpContext(int Value, Boolean ByCommand);
```

Description

Call GetHelpContext to obtain the ID for context-sensitive help on a particular menu item. Identify the menu item as the item that matches the Value property. If ByCommand is true, Value is the Command property of the TMenuItem (see page 1363). If ByCommand is false, Value is the Handle (see page 1345) property.

See Also

FindItem (see page 1337)

Command (see page 1401)

Handle (see page 1409)

HelpContext (see page 1410)

HelpContext

34.1.3.4.12 TMenu.IsOwnerDraw Method

Indicates whether menu items can contain more than simple captions.

Pascal

```
function IsOwnerDraw: Boolean;
```

C++

```
__fastcall Boolean IsOwnerDraw();
```

Description

IsOwnerDraw returns true when the OwnerDraw (see page 1350) property is true or the Images (see page 1345) property is not nil (Delphi) or NULL (C++). This is used internally to determine how the menu should be painted.

See Also

OwnerDraw (see page 1350)

Images (see page 1345)

34.1.3.4.13 TMenu.IsRightToLeft Method

Indicates whether menu items read from right to left.

Pascal

```
function IsRightToLeft: Boolean;
```

C++

```
__fastcall Boolean IsRightToLeft();
```

Description

Call IsRightToLeft to determine the layout of main menu items. IsRightToLeft returns true if the application is running on a system with a Middle Eastern locale and the BiDiMode (see page 1344) property is not bdLeftToRight.

See Also

BiDiMode (see page 1344)

34.1.3.4.14 TMenu.IsShortCut Method

Dispatches a key message if it represents a shortcut key for an item in the menu.

Pascal

```
function IsShortCut(var Message: TWMKey): Boolean; dynamic;
```

C++

```
__fastcall Boolean IsShortCut(TWMKey Message);
```

Description

Call IsShortCut when processing a KeyDown message to identify shortcut keys and trigger the appropriate menu action. IsShortCut decodes the key message passed in the Message parameter. It then encodes the state of the Shift, Control, and Alt keys with the value of the character that was typed to obtain the ShortCut (see page 1453) property of a menu item. If IsShortCut locates a menu item associated with the shortcut key combination, it calls the item's Click method and returns true. If no item in the menu has a shortcut corresponding to the message, IsShortCut returns false.

See Also

DispatchCommand (see page 1335)

DispatchPopup (see page 1336)

FindItem (see page 1337)

Click (see page 1375)

34.1.3.4.15 TMenu.Loaded Method

Calls the DoChange (see page 1336) method when the menu is first loaded into memory.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

Loaded is called automatically immediately after the menu is loaded into memory. It calls the DoChange (see page 1336) method to allow any necessary fixup.

See Also

DoChange (see page 1336)

34.1.3.4.16 TMenu.MenuChanged Method

Responds when the menu changes.

Pascal

```
procedure MenuChanged(Sender: TObject; Source: TMenuItem; Rebuild: Boolean); virtual;
```

C++

```
virtual __fastcall MenuChanged(TObject * Sender, TMenuItem Source, Boolean Rebuild);
```

Description

MenuChanged is called automatically when changes occur that affect the structure of the menu. The Sender parameter indicates the object that originates the change. It's value can be nil (Delphi) or NULL (C++). Source indicates which menu item is affected. Rebuild indicates whether the menu must be rebuilt to reflect the change (for example when items are added or deleted).

MenuChanged calls the DoChange (see page 1336) method, which generates an OnChange (see page 1352) event.

Note: Changes that affect individual items rather than the menu as a whole trigger the UpdateItems (see page 1343) method instead of MenuChanged. For example, when the Images (see page 1345) property changes, UpdateItems (see page 1343) is called rather than MenuChanged.

See Also

Items (see page 1346)

UpdateItems (see page 1343)

DoChange (see page 1336)

34.1.3.4.17 TMenu.Notification Method

Responds when components are added or deleted.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

TMenu (see page 1331) overrides the inherited method to set the Images (see page 1345) property to nil (Delphi) or NULL (C++) if the ImageList is deleted.

See Also

Notification

Images (see page 1345)

34.1.3.4.18 ParentBiDiModeChanged Method

This is the overview for the ParentBiDiModeChanged method overload.

Overload List

	Name	Description
	TMenu.ParentBiDiModeChanged () (see page 1341)	Ensures that the menu's layout follows the current settings of the BiDiMode (see page 1344) and ParentBiDiMode (see page 1351) properties.
	TMenu.ParentBiDiModeChanged (TObject) (see page 1341)	Ensures that the menu's layout follows the current settings of the BiDiMode (see page 1344) and ParentBiDiMode (see page 1351) properties.

34.1.3.4.18.1 TMenu.ParentBiDiModeChanged Method ()

Ensures that the menu's layout follows the current settings of the BiDiMode (see page 1344) and ParentBiDiMode (see page 1351) properties.

Pascal

```
procedure ParentBiDiModeChanged; overload;
procedure ParentBiDiModeChanged(AControl: TObject); overload;
```

C++

```
__fastcall ParentBiDiModeChanged();
__fastcall ParentBiDiModeChanged(TObject * AControl);
```

Description

ParentBiDiMode (see page 1351) changed is called when the menu's BiDiMode (see page 1344) property must be updated to reflect the bi-directional mode of its associated control.

See Also

BiDiMode (see page 1344)

ParentBiDiMode (see page 1351)

DoBiDiModeChanged (see page 1336)

34.1.3.4.19 TMenu.ProcessMenuChar Method

Responds to keyboard input for the menu.

Pascal

```
procedure ProcessMenuChar(var Message: TWMMenuChar);
```

C++

```
__fastcall ProcessMenuChar(TWMMenuChar Message);
```

Description

ProcessMenuChar handles keyboard messages from Windows. It checks whether the character typed is an accelerator character from one of the menu's items. If so, ProcessMenuChar executes the menu item (or, if the item is disabled, selects it). If the key is not an accelerator, ProcessMenuChar checks for captions that begin with the character. If there is only one, it is treated like an accelerator. Otherwise, the first item beginning with that character is selected.

When calling ProcessMenuChar programmatically, the User, MenuFlag, and Menu fields of Message parameter must be supplied.

See Also

Caption (see page 1396)

34.1.3.4.20 TMenu.SetChildOrder Method

Ensures that the menu item indexes reflect the order in which they are loaded and saved.

Pascal

```
procedure SetChildOrder(Child: TComponent; Order: Integer); override;
```

C++

```
virtual __fastcall SetChildOrder(TComponent * Child, int Order);
```

Description

SetChildOrder is called automatically when menus are loaded into memory. The Child parameter specifies the child object that was just loaded. The Order parameter specifies the order in which it was loaded. SetChildOrder assigns the MenuIndex for each child to match the order in which the items are loaded.

See Also

MenuIndex (see page 1413)

34.1.3.4.21 TMenu.UpdateItems Method

Causes all menu items to update themselves.

Pascal

```
procedure UpdateItems;
```

C++

```
__fastcall UpdateItems();
```

Description

UpdateItems is called automatically when changes occur that must be reflected by individual menu items. Changes that affect the menu as a whole trigger the MenuChanged (see page 1340) method instead.

See Also

MenuChanged (see page 1340)

34.1.3.5 TMenu Properties

The properties of the TMenu class are listed here.

Public Properties

	Name	Description
	AutoHotkeys (see page 1343)	Determines whether the accelerator keys for menu items can be reset automatically.
	AutoLineReduction (see page 1344)	Determines whether redundant separator bars are automatically removed from the menu.
	BiDiMode (see page 1344)	Determines the layout of the menu in Middle East locales.
R	Handle (see page 1345)	Provides access to the Windows menu handle for the menu.
	Images (see page 1345)	Lists the images that can appear beside individual menu items.
	OwnerDraw (see page 1350)	Specifies whether the menu items are drawn by the application.
	ParentBiDiMode (see page 1351)	Indicates whether the BiDiMode (see page 1344) property is inherited from the control that uses the menu.
	WindowHandle (see page 1351)	Provides access to the handle of the window that uses the menu.

Published Properties

	Name	Description
R	Items (see page 1346)	Describes the elements of the menu.

34.1.3.5.1 TMenu.AutoHotkeys Property

Determines whether the accelerator keys for menu items can be reset automatically.

Pascal

```
property AutoHotkeys: TMenuAutoFlag;
```

C++

```
__property TMenuAutoFlag AutoHotkeys;
```

Description

Set AutoHotkeys to indicate whether the menu should automatically ensure that accelerator keys for its items are reset if necessary so that every menu item has an accelerator key and no two items have the same accelerator key. This ensures that menu items added dynamically at runtime can have accelerator keys that do not introduce conflicts. TMenu (see page 1331)

tries to preserve all existing accelerator key mappings when adjusting the accelerator keys, but may be forced to change some in order to ensure that every item has a unique accelerator.

Before the menu is displayed, the menu checks and adjusts the accelerator key mappings if AutoHotkeys is maAutomatic. When AutoHotkeys is maManual, the menu does not alter the top-level menu items, but may adjust the accelerator keys for submenus, depending on the AutoHotkeys property of the menu item that is the parent of the submenu.

When AutoHotkeys is maManual, an application can still adjust the menu items by calling the Items (see page 1346) property's RethinkHotkeys method. AutoHotkeys merely controls whether this adjustment occurs automatically.

Note: The AutoHotkeys property of a menu is linked to the AutoHotkeys property of the TMenuItems object that implements its Items (see page 1346) property. Setting one sets the other.

See Also

AutoHotkeys (see page 1394)

Items (see page 1346)

RethinkHotkeys (see page 1388)

34.1.3.5.2 TMenu.AutoLineReduction Property

Determines whether redundant separator bars are automatically removed from the menu.

Pascal

```
property AutoLineReduction: TMenuAutoFlag;
```

C++

```
__property TMenuAutoFlag AutoLineReduction;
```

Description

Set AutoLineReduction to indicate whether the menu should automatically ensure that the menu does not start or end with a separator bar and that two separator bars do not appear next to each other. This ensures that menus built dynamically at runtime do not have separator bars in undesired positions.

Before the menu is displayed, it removes superfluous separator bars when AutoLineReduction is maAutomatic. When AutoLineReduction is maManual, the menu does not remove separators from the top-level menu items, but may remove superfluous separators from submenus, depending on the AutoLineReduction property of the menu item that is the parent of the submenu.

When AutoLineReduction is maManual, an application can still remove superfluous separators by calling the Items (see page 1346) property's RethinkLines method. AutoLineReduction merely determines whether this adjustment occurs automatically.

Note: The AutoLineReduction property of a menu is linked to the AutoLineReduction property of the TMenuItems object that implements its Items (see page 1346) property. Setting one sets the other.

See Also

AutoLineReduction (see page 1395)

Items (see page 1346)

RethinkLines (see page 1389)

34.1.3.5.3 TMenu.BiDiMode Property

Determines the layout of the menu in Middle East locales.

Pascal

```
property BiDiMode: TBiDiMode;
```

C++

```
__property TBiDiMode BiDiMode;
```

Description

Set BiDiMode to control the layout of menu items. BiDiMode controls whether the menu items read from left to right or right to left.

See Also

ParentBiDiMode (see page 1351)

IsRightToLeft (see page 1339)

34.1.3.5.4 TMenu.Handle Property

Provides access to the Windows menu handle for the menu.

Pascal

```
property Handle: HMENU;
```

C++

```
__property HMENU Handle;
```

Description

Use Handle to call a Windows API function that requires a menu handle. Handle is the handle for the menu that contains only the items described by the Items (see page 1346) property. If no other menus have been merged with this menu, this is the same as the WindowHandle (see page 1351) property.

See Also

WindowHandle (see page 1351)

34.1.3.5.5 TMenu.Images Property

Lists the images that can appear beside individual menu items.

Pascal

```
property Images: TCustomImageList;
```

C++

```
__property TCustomImageList Images;
```

Description

Use Images to hold the set of images that can appear beside menu items in the menu. Individual menu items set their ImageIndex property to indicate which image in the list appears to the left of the menu item caption.

Images in the list should be 16x16.

Tip: If the application uses action lists, this property should be the value of the action list's Images property. This way, menu items and buttons that do the same thing can display the same image.

See Also

ImageIndex (see page 1411)

Images

Items (see page 1346)

34.1.3.5.6 TMenu.Items Property

Describes the elements of the menu.

Pascal

```
property Items: TMenuItem;
```

C++

```
__property TMenuItem Items;
```

Description

Use Items to access information about the elements in the menu. Items is a single TMenuItem (see page 1363) object that describes the elements of the menu in its own Items property.

Note: Because Items is the default property of TMenuItem (see page 1363), the Items property of TMenu (see page 1331) can be treated as an indexed array of menu items that describe the individual items in the menu. That is, instead of writing

Note: FirstItem := Menu1.Items.Items[0];

Note: you can use

Note: FirstItem := Menu1.Items[0];

Set the Items property at design time by clicking on the Items property in the Property Inspector. The Menu Designer automatically inserts a menu item. As each menu item is entered, outlined regions appear at the possible locations of additional menu items. Click in those regions to specify additional menu items. At runtime, create a TMenuItem (see page 1363) that describes the elements of the menu programmatically, and set the Items property of the menu.

C++ Examples:

```
/*
This example assumes that the main form of the application
has a main menu with a menu item. The following code adds a
separator, and the name of all forms to the menu. Do not
access FormCount in the mainform's FormCreate.  The other
forms do not exist yet!
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMenuItem *NewItem;
  // first create the separator
  NewItem = new TMenuItem(this); // If you change the AOwner (this) to MainMenu1->Items[1] and
then you add the new menu you will get "Menu inserted twice"?  Doesn't happen in Delphi.
  NewItem->Name = "Separator";   // The owner will cleanup these menu items.
  NewItem->Caption = "-";
  // add the new item to the Windows menu
  MainMenu1->Items->Items[1]->Add(NewItem);
  // now create and add a menu item for each form
  for  (int I = 0; I < Screen->FormCount; I++)
  {
    NewItem = new TMenuItem(this); // The owner will cleanup these menu items.
    NewItem->Caption = Screen->Forms[I]->Name + "Item";
    NewItem->Name = Screen->Forms[I]->Name;
    MainMenu1->Items->Items[1]->Add(NewItem);
```

```
  }
}
/*
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
    {
        switch (ClickedItem->Tag)
        {
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
            }
        }
    }
}

/*
To edit or destroy an item, grab its pointer via the Items
property.
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}
```

```
void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
This example assumes that the main form of the application
has a main menu with a menu item. The following code adds a
separator, and the name of all forms to the menu. Do not
access FormCount in the mainform's FormCreate.  The other
forms do not exist yet!
}
procedure TForm1.Button1Click(Sender: TObject);
var
  NewItem: TMenuItem;
  I : integer;
begin
  { first create the separator }
  NewItem := TMenuItem.Create(MainMenu1.Items[1]);
  NewItem.Name:= 'Separator';
  NewItem.Caption := '-';
  { add the new item to the Windows menu }
  MainMenu1.Items[1].Add(NewItem);
  { now create and add a menu item for each form }
  for  I := 0 to Screen.FormCount-1 do
  begin
    NewItem := TMenuItem.Create(MainMenu1.Items[1]);
    NewItem.Caption := Screen.Forms[I].Name + 'Item';
    NewItem.Name := Screen.Forms[I].Name;
    MainMenu1.Items[1].Add(NewItem);
   end;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  I : Integer;
begin
  // Count down, not up!
  for  I := MainMenu1.Items[1].Count - 1 downto 0 do
  begin
    MainMenu1.Items[1].Items[I].Free;
  end;
end;
{
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
```

```
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
}

type
  TForm1 = class(TForm)
    AddButton: TButton;
    EditButton: TButton;
    DestroyButton: TButton;
    PopupMenu1: TPopupMenu;
    procedure AddButtonClick(Sender: TObject);
    procedure EditButtonClick(Sender: TObject);
    procedure DestroyButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    procedure PopupMenuItemsClick(Sender: TObject);

  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.AddButtonClick(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
    // The owner (PopupMenu1) will cleanup this menu item.
  NewItem := TMenuItem.Create(PopupMenu1); // create the new item
  index := PopupMenu1.Items.Count;
  PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
  NewItem.Caption := 'Menu Item ' + IntToStr(index);
  NewItem.Tag := index;
  NewItem.OnClick :=
    PopupMenuItemsClick; // assign it an event handler
end;

procedure TForm1.PopupMenuItemsClick(Sender: TObject);
begin
  with Sender as TMenuItem do
  begin
    case Tag of
      0:  ShowMessage('first item clicked');
      1:  ShowMessage('second item clicked');
      2:  ShowMessage('third item clicked');
      3:  ShowMessage('fourth item clicked');
    end;
  end;
end;

{
To edit or destroy an item, grab its pointer via the Items
property.
}
procedure TForm1.EditButtonClick(Sender: TObject);
var
```

```
  ItemToEdit: TMenuItem;
begin
  ItemToEdit := PopupMenu.Items[1];
  ItemToEdit.Caption := 'Changed Caption';
end;

procedure TForm1.DestroyButtonClick(Sender: TObject);
var
  ItemToDelete: TMenuItem;
begin
  ItemToDelete := PopupMenu.Items[2];
  ItemToDelete.Free;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
  for index := 0 to 3 do
  begin
      // The owner (PopupMenu1) will cleanup this menu item.
    NewItem := TMenuItem.Create(PopupMenu1); // create the new item
    PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
    NewItem.Caption := 'Menu Item ' + IntToStr(index);
    NewItem.Tag := index;
    NewItem.OnClick :=
      PopupMenuItemsClick; // assign it an event handler
  end;
end;
```

See Also

TMenuItem (see page 1363)

Items (see page 1411)

34.1.3.5.7 TMenu.OwnerDraw Property

Specifies whether the menu items are drawn by the application.

Pascal

```
property OwnerDraw: Boolean;
```

C++

```
__property Boolean OwnerDraw;
```

Description

Set OwnerDraw to true to indicate that the application will draw menu items in an OnDrawItem event handler. When OwnerDraw is true, menu items receive an OnMeasureItem and an OnDrawItem event when they need to be rendered on screen.

Set OwnerDraw to false to accept the default drawing. When OwnerDraw is false, menu items display the caption with an optional image to the left (If the Images (see page 1345) and ImageIndex properties are set).

Owner-draw menus are useful for displaying the value that appears when selecting a menu item. For example, in a menu that allows users to choose a color, an owner-draw menu could display rectangles drawn in the designated color, rather than color names.

See Also

OnMeasureItem (see page 1423)

OnDrawItem (see page 1422)

Images (see page 1345)

34.1.3.5.8 TMenu.ParentBiDiMode Property

Indicates whether the BiDiMode (see page 1344) property is inherited from the control that uses the menu.

Pascal

```
property ParentBiDiMode: Boolean;
```

C++

```
__property Boolean ParentBiDiMode;
```

Description

By default, ParentBiDiMode is true, indicating that the layout of menu items follows the BiDiMode (see page 1344) setting of the associated form or control. Change ParentBiDiMode to false to override the default BiDiMode (see page 1344) setting and control the menu's layout independently.

See Also

BiDiMode (see page 1344)

34.1.3.5.9 TMenu.WindowHandle Property

Provides access to the handle of the window that uses the menu.

Pascal

```
property WindowHandle: HWND;
```

C++

```
__property HWND WindowHandle;
```

Description

Read WindowHandle to get the window handle of the control that receives WM_COMMAND (see page 1684) messages when the user clicks on an item in the menu. When menu items are selected, use this handle in an OnClick event handler to send messages to the control that uses the menu or to make Windows API calls that affect the control.

Set WindowHandle to associate the menu with a control.

See Also

DispatchCommand (see page 1335)

Command (see page 1401)

OnClick (see page 1417)

34.1.3.6 TMenu Events

The events of the TMenu class are listed here.

Protected Events

	Name	Description
	OnChange (see page 1352)	Occurs when the menu changes.

34.1.3.6.1 TMenu.OnChange Event

Occurs when the menu changes.

Pascal

```
property OnChange: TMenuChangeEvent;
```

C++

```
__property TMenuChangeEvent OnChange;
```

Description

Write an OnChange event handler to respond to changes in the menu's structure. OnChange occurs when the menu is loaded into memory, and in response to property changes that affect the menu's structure.

See Also

Items (see page 1346)

34.1.4 TMenuActionLink Class

TMenuActionLink links menu item clients to the corresponding menu action.

Class Hierarchy

Pascal

```
TMenuActionLink = class(TActionLink);
```

C++

```
class TMenuActionLink : public TActionLink;
```

File

Menus

Description

TMenuActionLink is designed to link actions to menu item clients. TMenuActionLink overrides methods to determine whether the action and client are linked, and to ensure that action events are appropriate for properties of and events of menu item clients.

TMenuActionLink overrides methods to support linking the following properties and events with menu item client properties or events, most of which have the same name:

Caption

Checked

Enabled

HelpContext

Hint

ImageIndex

ShortCut (see page 1453)

Visible

OnExecute

34.1.4.1 TMenuActionLink Members

The following tables list the members exposed by TMenuActionLink.

Protected Fields

	Name	Description
	FClient (see page 1354)	This is FClient, a member of class TMenuActionLink.

Protected Methods

	Name	Description
	AssignClient (see page 1354)	Assigns a client menu item that is linked to the associated action.
	IsAutoCheckLinked (see page 1355)	Indicates whether the AutoCheck state for the action and the client are linked.
	IsCaptionLinked (see page 1355)	Indicates whether the caption for the action and the client are linked.
	IsCheckedLinked (see page 1356)	Indicates whether the checked state for the action and the client are linked.
	IsEnabledLinked (see page 1356)	Indicates whether the enabled state for the action and the client are linked.
	IsGroupIndexLinked (see page 1356)	Indicates whether the GroupIndex property for the action and the client are linked.
	IsHelpContextLinked (see page 1357)	Indicates whether the HelpContext properties for the action and client controls are linked.
	IsHintLinked (see page 1357)	Indicates whether the Hint property for the action and the client are linked.
	IsImageIndexLinked (see page 1358)	Indicates whether the image index for the action and the client are linked.
	IsOnExecuteLinked (see page 1358)	Indicates whether the OnExecute event handler for the action and the client are linked.
	IsShortCutLinked (see page 1358)	Indicates whether the ShortCut (see page 1453) property for the action and the client are linked.
	IsVisibleLinked (see page 1359)	Indicates whether the visible state for the action and the client are linked.
	SetAutoCheck (see page 1359)	Sets the AutoCheck property of a menu item if the client is linked to the action for this property.
	SetCaption (see page 1359)	Sets the caption of a menu item if the client is linked to the action for this property.
	SetChecked (see page 1360)	Sets the checked state of a menu item if the client is linked to the action for this property.
	SetEnabled (see page 1360)	Sets the enabled state of a menu item if the client is linked to the action for this property.
	SetHelpContext (see page 1361)	Sets the Help context of a menu item that is the client of this action link.
	SetHint (see page 1361)	Sets the hint of a menu item if the client is linked to the action for this property.
	SetImageIndex (see page 1361)	Sets the image index of a menu item if the client is linked to the action for this property.
	SetOnExecute (see page 1362)	Sets the execution event handler for a menu item if the client is linked to the action for this property.
	SetShortCut (see page 1362)	Sets the shortcut of a menu item if the client is linked to the action for this property.
	SetVisible (see page 1362)	Sets the visible state of a menu item if the client is linked to the action for this property.

34.1.4.2 TMenuActionLink Fields

The fields of the TMenuActionLink class are listed here.

Protected Fields

	Name	Description
	FClient (see page 1354)	This is FClient, a member of class TMenuActionLink.

34.1.4.2.1 TMenuActionLink.FClient Field

Pascal

```
FClient: TMenuItem;
```

C++

```
TMenuItem FClient;
```

Description

This is FClient, a member of class TMenuActionLink.

34.1.4.3 TMenuActionLink Methods

The methods of the TMenuActionLink class are listed here.

Protected Methods

	Name	Description
	AssignClient (see page 1354)	Assigns a client menu item that is linked to the associated action.
	IsAutoCheckLinked (see page 1355)	Indicates whether the AutoCheck state for the action and the client are linked.
	IsCaptionLinked (see page 1355)	Indicates whether the caption for the action and the client are linked.
	IsCheckedLinked (see page 1356)	Indicates whether the checked state for the action and the client are linked.
	IsEnabledLinked (see page 1356)	Indicates whether the enabled state for the action and the client are linked.
	IsGroupIndexLinked (see page 1356)	Indicates whether the GroupIndex property for the action and the client are linked.
	IsHelpContextLinked (see page 1357)	Indicates whether the HelpContext properties for the action and client controls are linked.
	IsHintLinked (see page 1357)	Indicates whether the Hint property for the action and the client are linked.
	IsImageIndexLinked (see page 1358)	Indicates whether the image index for the action and the client are linked.
	IsOnExecuteLinked (see page 1358)	Indicates whether the OnExecute event handler for the action and the client are linked.
	IsShortCutLinked (see page 1358)	Indicates whether the ShortCut (see page 1453) property for the action and the client are linked.
	IsVisibleLinked (see page 1359)	Indicates whether the visible state for the action and the client are linked.
	SetAutoCheck (see page 1359)	Sets the AutoCheck property of a menu item if the client is linked to the action for this property.
	SetCaption (see page 1359)	Sets the caption of a menu item if the client is linked to the action for this property.
	SetChecked (see page 1360)	Sets the checked state of a menu item if the client is linked to the action for this property.
	SetEnabled (see page 1360)	Sets the enabled state of a menu item if the client is linked to the action for this property.
	SetHelpContext (see page 1361)	Sets the Help context of a menu item that is the client of this action link.
	SetHint (see page 1361)	Sets the hint of a menu item if the client is linked to the action for this property.
	SetImageIndex (see page 1361)	Sets the image index of a menu item if the client is linked to the action for this property.
	SetOnExecute (see page 1362)	Sets the execution event handler for a menu item if the client is linked to the action for this property.
	SetShortCut (see page 1362)	Sets the shortcut of a menu item if the client is linked to the action for this property.
	SetVisible (see page 1362)	Sets the visible state of a menu item if the client is linked to the action for this property.

34.1.4.3.1 TMenuActionLink.AssignClient Method

Assigns a client menu item that is linked to the associated action.

Pascal

```
procedure AssignClient(AClient: TObject); override;
```

C++

```
virtual __fastcall AssignClient(TObject * AClient);
```

Description

TMenuActionLink (see page 1352) overrides AssignClient to ensure that the client for the link is a menu item.

See Also

Action (see page 1393)

Create

34.1.4.3.2 TMenuActionLink.IsAutoCheckLinked Method

Indicates whether the AutoCheck state for the action and the client are linked.

Pascal

```
function IsAutoCheckLinked: Boolean; virtual;
```

C++

```
virtual __fastcall Boolean IsAutoCheckLinked();
```

Description

IsAutoCheckLinked indicates whether the AutoCheck property is linked to the appropriate client property. IsAutoCheckLinked returns true if the client requests that the action control this property, false otherwise.

See Also

AutoCheck

AutoCheck (see page 1394)

SetAutoCheck (see page 1359)

34.1.4.3.3 TMenuActionLink.IsCaptionLinked Method

Indicates whether the caption for the action and the client are linked.

Pascal

```
function IsCaptionLinked: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsCaptionLinked();
```

Description

IsCaptionLinked indicates whether the Caption property is linked to the appropriate client property. IsCaptionLinked returns true if the client requests that the action control this property, false otherwise.

See Also

Caption

Caption (see page 1396)

SetCaption (see page 1359)

34.1.4.3.4 TMenuActionLink.IsCheckedLinked Method

Indicates whether the checked state for the action and the client are linked.

Pascal

```
function IsCheckedLinked: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsCheckedLinked();
```

Description

IsCheckedLinked indicates whether the Checked property is linked to the appropriate client property. IsCheckedLinked returns true if the client requests that the action control this property, false otherwise.

See Also

Checked

Checked (see page 1401)

SetChecked (see page 1360)

34.1.4.3.5 TMenuActionLink.IsEnabledLinked Method

Indicates whether the enabled state for the action and the client are linked.

Pascal

```
function IsEnabledLinked: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsEnabledLinked();
```

Description

IsEnabledLinked indicates whether the Enabled property is linked to the appropriate client property. IsEnabledLinked returns true if the client requests that the action control this property, false otherwise.

See Also

Enabled

Enabled (see page 1406)

SetEnabled (see page 1360)

34.1.4.3.6 TMenuActionLink.IsGroupIndexLinked Method

Indicates whether the GroupIndex property for the action and the client are linked.

Pascal

```
function IsGroupIndexLinked: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsGroupIndexLinked();
```

Description

IsGroupIndexLinked indicates whether the GroupIndex property is linked to the menu item's GroupIndex property. IsGroupIndexLinked returns true if the client menu item has a RadioItem property of true and the client requests that the action

control this property. It returns false if the client menu item has a RadioItem property of false or has severed its GroupIndex property from the action.

See Also

RadioItem (see page 1414)

GroupIndex (see page 1408)

34.1.4.3.7 TMenuActionLink.IsHelpContextLinked Method

Indicates whether the HelpContext properties for the action and client controls are linked.

Pascal

```
function IsHelpContextLinked: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsHelpContextLinked();
```

Description

TActionLink descendants can call IsHelpContextLinked to determine whether they need to propagate changes to the HelpContext property of the action so that they are reflected by the client object. When IsHelpContextLinked returns true, the action controls the HelpContext property: Changes to the HelpContext property of the action are reflected in the client object. When IsHelpContextLinked returns false, the client object is not influenced by the HelpContext property of its action.

As implemented in TActionLink, IsHelpContextLinked merely checks that the Action property specifies a descendant of TCustomAction, and if so, returns true.

See Also

HelpContext

SetHelpContext

Action

IsHelpLinked

34.1.4.3.8 TMenuActionLink.IsHintLinked Method

Indicates whether the Hint property for the action and the client are linked.

Pascal

```
function IsHintLinked: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsHintLinked();
```

Description

IsHintLinked indicates whether the Hint property is linked to the appropriate client property. IsHintLinked returns true if the client requests that the action control this property, false otherwise.

See Also

Hint

Hint (see page 1410)

SetHint (see page 1361)

34.1.4.3.9 TMenuActionLink.IsImageIndexLinked Method

Indicates whether the image index for the action and the client are linked.

Pascal

```
function IsImageIndexLinked: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsImageIndexLinked();
```

Description

IsImageIndexLinked indicates whether the ImageIndex property is linked to the appropriate client property. IsImageIndexLinked returns true if the client requests that the action control this property, false otherwise.

See Also

ImageIndex

ImageIndex (see page 1411)

SetImageIndex (see page 1361)

34.1.4.3.10 TMenuActionLink.IsOnExecuteLinked Method

Indicates whether the OnExecute event handler for the action and the client are linked.

Pascal

```
function IsOnExecuteLinked: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsOnExecuteLinked();
```

Description

IsOnExecuteLinked indicates whether the OnExecute event handler is linked to the appropriate client event. IsOnExecuteLinked returns true if the client requests that the action control this event, false otherwise.

See Also

OnExecute

OnClick (see page 1417)

SetOnExecute (see page 1362)

34.1.4.3.11 TMenuActionLink.IsShortCutLinked Method

Indicates whether the ShortCut (see page 1453) property for the action and the client are linked.

Pascal

```
function IsShortCutLinked: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsShortCutLinked();
```

Description

IsShortCutLinked indicates whether the ShortCut (see page 1453) property is linked to the appropriate client property. IsVisibleLinked (see page 1359) returns true if the client requests that the action control this property, false otherwise.

See Also

ShortCut

ShortCut (see page 1415)

SetShortCut (see page 1362)

34.1.4.3.12 TMenuActionLink.IsVisibleLinked Method

Indicates whether the visible state for the action and the client are linked.

Pascal

```
function IsVisibleLinked: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsVisibleLinked();
```

Description

IsVisibleLinked indicates whether the Visible property is linked to the appropriate client property. IsVisibleLinked returns true if the client requests that the action control this property, false otherwise.

See Also

Visible

Visible (see page 1416)

SetVisible (see page 1362)

34.1.4.3.13 TMenuActionLink.SetAutoCheck Method

Sets the AutoCheck property of a menu item if the client is linked to the action for this property.

Pascal

```
procedure SetAutoCheck(Value: Boolean); override;
```

C++

```
virtual __fastcall SetAutoCheck(Boolean Value);
```

Description

SetAutoCheck sets the AutoCheck property of a menu item, and is called only if the client is linked for this event. Value specifies the AutoCheck new value of AutoCheck.

See Also

AutoCheck

AutoCheck (see page 1394)

IsAutoCheckLinked (see page 1355)

34.1.4.3.14 TMenuActionLink.SetCaption Method

Sets the caption of a menu item if the client is linked to the action for this property.

Pascal

```
procedure SetCaption(const Value: string); override;
```

C++

```
virtual __fastcall SetCaption(const UnicodeString Value);
```

Description

SetCaption sets the caption of a menu item, and is called only if the client is linked for this event. Value is the string to which the caption should be set.

See Also

Caption

Caption (see page 1396)

IsCaptionLinked (see page 1355)

34.1.4.3.15 TMenuActionLink.SetChecked Method

Sets the checked state of a menu item if the client is linked to the action for this property.

Pascal

```
procedure SetChecked(Value: Boolean); override;
```

C++

```
virtual __fastcall SetChecked(Boolean Value);
```

Description

SetChecked sets the checked state of a menu item, and is called only if the client is linked for this event. Value specifies whether the checked state should be set.

See Also

Checked

Checked (see page 1401)

IsCheckedLinked (see page 1356)

34.1.4.3.16 TMenuActionLink.SetEnabled Method

Sets the enabled state of a menu item if the client is linked to the action for this property.

Pascal

```
procedure SetEnabled(Value: Boolean); override;
```

C++

```
virtual __fastcall SetEnabled(Boolean Value);
```

Description

SetEnabled sets the enabled state of a menu item, and is called only if the client is linked for this event. Value specifies whether the enabled state should be set.

See Also

Enabled

Enabled (see page 1406)

IsEnabledLinked (see page 1356)

34.1.4.3.17 TMenuActionLink.SetHelpContext Method

Sets the Help context of a menu item that is the client of this action link.

Pascal

```
procedure SetHelpContext(Value: THelpContext); override;
```

C++

```
virtual __fastcall SetHelpContext(THelpContext Value);
```

Description

SetHelpContext sets the help context of a client menu item, and is called only if the client is linked for this property. Value specifies what the help context should be.

See Also

HelpContext

HelpContext (see page 1410)

IsHelpLinked

34.1.4.3.18 TMenuActionLink.SetHint Method

Sets the hint of a menu item if the client is linked to the action for this property.

Pascal

```
procedure SetHint(const Value: string); override;
```

C++

```
virtual __fastcall SetHint(const UnicodeString Value);
```

Description

SetHint sets the hint of a menu item, and is called only if the client is linked for this event. Value is the string that to which the hint should be set.

See Also

Hint

Hint (see page 1410)

IsHintLinked (see page 1357)

34.1.4.3.19 TMenuActionLink.SetImageIndex Method

Sets the image index of a menu item if the client is linked to the action for this property.

Pascal

```
procedure SetImageIndex(Value: Integer); override;
```

C++

```
virtual __fastcall SetImageIndex(int Value);
```

Description

SetImageIndex sets the image index of a menu item, and is called only if the client is linked for this event. Value specifies what the image index should be.

See Also

ImageIndex

ImageIndex (see page 1411)

IsImageIndexLinked (see page 1358)

34.1.4.3.20 TMenuActionLink.SetOnExecute Method

Sets the execution event handler for a menu item if the client is linked to the action for this property.

Pascal

```
procedure SetOnExecute(Value: TNotifyEvent); override;
```

C++

```
virtual __fastcall SetOnExecute(TNotifyEvent Value);
```

Description

SetOnExecute sets the appropriate execution event handler of a menu item, and is called only if the client is linked for this event. Value specifies the procedure that should be used for this event.

See Also

OnExecute

OnClick (see page 1417)

IsOnExecuteLinked (see page 1358)

34.1.4.3.21 TMenuActionLink.SetShortCut Method

Sets the shortcut of a menu item if the client is linked to the action for this property.

Pascal

```
procedure SetShortCut(Value: TShortCut); override;
```

C++

```
virtual __fastcall SetShortCut(TShortCut Value);
```

Description

SetShortCut sets the shortcut of a menu item, and is called only if the client is linked for this event. Value specifies what the shortcut should be.

See Also

ShortCut

ShortCut (see page 1415)

IsShortCutLinked (see page 1358)

34.1.4.3.22 TMenuActionLink.SetVisible Method

Sets the visible state of a menu item if the client is linked to the action for this property.

Pascal

```
procedure SetVisible(Value: Boolean); override;
```

C++

```
virtual __fastcall SetVisible(Boolean Value);
```

Description

SetVisible sets the visible state of a menu item, and is called only if the client is linked for this event. Value specifies whether the visible state should be set.

See Also

Visible

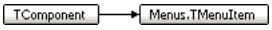
Visible (see page 1416)

IsVisibleLinked (see page 1359)

34.1.5 TMenuItem Class

TMenuItem describes the properties of an item in a menu.

Class Hierarchy



Pascal

```
TMenuItem = class(TComponent);
```

C++

```
class TMenuItem : public TComponent;
```

File

Menus

Description

Use TMenuItem to specify the appearance and behavior of an item in a menu. Each TMainMenu (see page 1322) or TPopupMenu (see page 1430) component can contain multiple menu items. When a menu is defined at design time, the Menu Designer automatically creates menu item objects for each command on the menu.

See Also

TMainMenu (see page 1322)

TPopupMenu (see page 1430)

NewItem (see page 1448)

NewLine (see page 1449)

NewSubMenu (see page 1452)

34.1.5.1 TMenuItem Members

The following tables list the members exposed by TMenuItem.

Protected Methods

	Name	Description
	ActionChange (see page 1369)	This is ActionChange, a member of class TMenuItem.
V	AdvancedDrawItem (see page 1374)	This is AdvancedDrawItem, a member of class TMenuItem.
V	AssignTo (see page 1374)	Copies the properties of an object to a destination object.
V	DefineProperties (see page 1375)	Designates methods for storing an object's unpublished data on a stream such as a form file.
	DoDrawText (see page 1376)	This is DoDrawText, a member of class TMenuItem.
V	DrawItem (see page 1377)	This is DrawItem, a member of class TMenuItem.
	GetActionLinkClass (see page 1377)	This is GetActionLinkClass, a member of class TMenuItem.
	GetAutoHotkeys (see page 1378)	This is GetAutoHotkeys, a member of class TMenuItem.
	GetAutoLineReduction (see page 1378)	This is GetAutoLineReduction, a member of class TMenuItem.
	GetCount (see page 1378)	This is GetCount, a member of class TMenuItem.
	GetHandle (see page 1379)	This is GetHandle, a member of class TMenuItem.
	GetItem (see page 1380)	This is GetItem, a member of class TMenuItem.
	GetMenuIndex (see page 1380)	This is GetMenuIndex, a member of class TMenuItem.
	InsertNewLine (see page 1383)	This is InsertNewLine, a member of class TMenuItem.
V	Loaded (see page 1385)	Initializes the component after the form file has been read into memory.
V	MeasureItem (see page 1386)	This is MeasureItem, a member of class TMenuItem.
V	MenuChanged (see page 1386)	This is MenuChanged, a member of class TMenuItem.
V	Notification (see page 1387)	Forwards notification messages to all owned components.
	SetBreak (see page 1389)	This is SetBreak, a member of class TMenuItem.
	SetCaption (see page 1389)	This is SetCaption, a member of class TMenuItem.
	SetChecked (see page 1390)	This is SetChecked, a member of class TMenuItem.
V	SetChildOrder (see page 1390)	Sets the visual ordering of a child control.
	SetDefault (see page 1390)	This is SetDefault, a member of class TMenuItem.
	SetEnabled (see page 1390)	This is SetEnabled, a member of class TMenuItem.
	SetGroupIndex (see page 1391)	This is SetGroupIndex, a member of class TMenuItem.
	SetImageIndex (see page 1391)	This is SetImageIndex, a member of class TMenuItem.
	SetMenuIndex (see page 1391)	This is SetMenuIndex, a member of class TMenuItem.
	SetRadioItem (see page 1392)	This is SetRadioItem, a member of class TMenuItem.
	SetShortCut (see page 1392)	This is SetShortCut, a member of class TMenuItem.
	SetVisible (see page 1392)	This is SetVisible, a member of class TMenuItem.
	UpdateItems (see page 1392)	This is UpdateItems, a member of class TMenuItem.

Protected Properties

	Name	Description
	ActionLink (see page 1394)	This is ActionLink, a member of class TMenuItem.

Public Methods

	Name	Description
	Add (see page 1369)	This is the overview for the Add method overload.
	Clear (see page 1375)	Removes and frees all menu items listed in the Items (see page 1411) property.
V	Click (see page 1375)	Simulates a mouse click.
V	Create (see page 1366)	Creates an instance of TMenuItem (see page 1363).
	Delete (see page 1376)	Removes a menu item from the Items (see page 1411) property array.
V	Destroy (see page 1367)	Destroys an instance of TMenuItem (see page 1363).
	Find (see page 1377)	Locates a menu item in the Items (see page 1411) property array given its caption.
V	GetChildren (see page 1378)	Enumerates all child components.
	GetEnumerator (see page 1379)	Returns a TComponent enumerator.
	GetImageList (see page 1379)	Returns the Image list that supplies an image for the menu item.
V	GetParentComponent (see page 1380)	Returns the parent component of the menu item.
	GetParentMenu (see page 1380)	Returns the main menu of which the menu item is part.

	HasParent (see page 1381)	Indicates that the menu item has a parent that is responsible for loading and saving its data.
	IndexOf (see page 1381)	Returns the position of a menu item within the Items (see page 1411) array.
	InitiateAction (see page 1382)	Calls the action link's Update method if the menu item is associated with an action link.
	Insert (see page 1382)	Inserts a menu item into a specified position in the Items (see page 1411) array.
	InsertNewLineAfter (see page 1383)	Inserts a separator bar after a specified item from the Items (see page 1411) property array.
	InsertNewLineBefore (see page 1384)	Inserts a separator bar before a specified item from the Items (see page 1411) property array.
	IsLine (see page 1385)	Indicates whether the menu item represents a separator bar.
	NewBottomLine (see page 1386)	Inserts a separator bar at the end of the Items (see page 1411) property array.
	NewTopLine (see page 1387)	Inserts a separator bar at the beginning of the Items (see page 1411) property array.
	Remove (see page 1388)	Removes a menu item from the Items (see page 1411) property array
	RethinkHotkeys (see page 1388)	Adjusts the captions of subitems so that every item has an accelerator and there are no duplicate accelerator keys.
	RethinkLines (see page 1389)	Removes superfluous separator lines from the Items (see page 1411) property array.
	SetParentComponent (see page 1391)	Sets the parent component.

Public Properties

	Name	Description
	Command (see page 1401)	Specifies the Windows Command ID associated with the menu item.
	Count (see page 1402)	Indicates the number of subitems of the menu item.
	Handle (see page 1409)	Indicates the Windows menu handle of the drop-down menu associated with the menu item.
	Items (see page 1411)	Lists the menu items in the submenu of the menu item.
	MenuIndex (see page 1413)	Indicates the index of the menu item within its parent menu.
	Parent (see page 1414)	Identifies the parent menu item of this menu item.

Published Events

	Name	Description
	OnAdvancedDrawItem (see page 1416)	Occurs when an owner-draw menu needs to be drawn.
	OnClick (see page 1417)	Occurs when the user clicks menu item.
	OnDrawItem (see page 1422)	Occurs when an owner-draw menu needs to be drawn.
	OnMeasureItem (see page 1423)	Occurs on owner-draw menu items to determine menu item size before the menu is drawn.

Published Properties

	Name	Description
	Action (see page 1393)	Designates the action associated with the menu item.
	AutoCheck (see page 1394)	Indicates whether the menu item's checked state toggles automatically when the item is clicked.
	AutoHotkeys (see page 1394)	Determines whether the accelerator keys for submenu items can be reset automatically.
	AutoLineReduction (see page 1395)	Determines whether redundant separator bars are automatically removed from the submenu.
	Bitmap (see page 1395)	Designates a bitmap that is drawn for the menu item.
	Break (see page 1396)	Determines whether the menu item starts a new column in the menu.
	Caption (see page 1396)	Specifies the text of the menu item.
	Checked (see page 1401)	Specifies whether a check mark should appear beside the Caption (see page 1396).
	Default (see page 1406)	Specifies whether the menu item is invoked when the parent item is double clicked.
	Enabled (see page 1406)	Specifies whether the menu item is enabled.
	GroupIndex (see page 1408)	Identifies the logical group to which the menu item belongs.
	HelpContext (see page 1410)	Specifies the help context ID associated with the menu item.

	Hint (see page 1410)	Specifies the text string that can appear when the user moves the mouse pointer over a menu item.
	ImageIndex (see page 1411)	Indicates which image maintained by the parent menu appears next to the menu item.
	RadioItem (see page 1414)	Specifies whether the menu item is mutually exclusive with other menu items in its group.
	ShortCut (see page 1415)	Specifies the key combination users can type to quickly access the menu item.
	SubMenuImages (see page 1415)	Lists the images that can appear beside individual items in a submenu of this menu item.
	Visible (see page 1416)	Specifies whether the menu item appears in the menu.

34.1.5.2 TMenuItem.Create Constructor

Creates an instance of TMenuItem (see page 1363).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TMenuItem(TComponent * AOwner);
```

Description

Use Create to create a menu item at runtime. Menu items defined at design time using the Menu designer are created automatically.

AOwner is the component that is responsible for freeing the menu item. It becomes the value of the Owner property.

C++ Examples:

```
/*
This example assumes that the main form of the application
has a main menu with a menu item. The following code adds a
separator, and the name of all forms to the menu. Do not
access FormCount in the mainform's FormCreate.  The other
forms do not exist yet!
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMenuItem *NewItem;
  // first create the separator
  NewItem = new TMenuItem(this); // If you change the AOwner (this) to MainMenu1->Items[1] and
then you add the new menu you will get "Menu inserted twice"?  Doesn't happen in Delphi.
  NewItem->Name = "Separator";   // The owner will cleanup these menu items.
  NewItem->Caption = "-";
  // add the new item to the Windows menu
  MainMenu1->Items->Items[1]->Add(NewItem);
  // now create and add a menu item for each form
  for  (int I = 0; I < Screen->FormCount; I++)
  {
    NewItem = new TMenuItem(this); // The owner will cleanup these menu items.
    NewItem->Caption = Screen->Forms[I]->Name + "Item";
    NewItem->Name = Screen->Forms[I]->Name;
    MainMenu1->Items->Items[1]->Add(NewItem);
  }
}
```

Delphi Examples:

```
{
This example assumes that the main form of the application
has a main menu with a menu item. The following code adds a
separator, and the name of all forms to the menu. Do not
```

```
  access FormCount in the mainform's FormCreate.  The other
  forms do not exist yet!
  }
  procedure TForm1.Button1Click(Sender: TObject);
  var
    NewItem: TMenuItem;
    I : integer;
  begin
    { first create the separator }
    NewItem := TMenuItem.Create(MainMenu1.Items[1]);
    NewItem.Name:= 'Separator';
    NewItem.Caption := '-';
    { add the new item to the Windows menu }
    MainMenu1.Items[1].Add(NewItem);
    { now create and add a menu item for each form }
    for  I := 0 to Screen.FormCount-1 do
    begin
      NewItem := TMenuItem.Create(MainMenu1.Items[1]);
      NewItem.Caption := Screen.Forms[I].Name + 'Item';
      NewItem.Name := Screen.Forms[I].Name;
      MainMenu1.Items[1].Add(NewItem);
     end;
  end;

  procedure TForm1.FormDestroy(Sender: TObject);
  var
    I : Integer;
  begin
    // Count down, not up!
    for  I := MainMenu1.Items[1].Count - 1 downto 0 do
    begin
      MainMenu1.Items[1].Items[I].Free;
    end;
  end;
```

See Also

Owner

NewItem (see page 1448)

34.1.5.3 TMenuItem.Destroy Destructor

Destroys an instance of TMenuItem (see page 1363).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TMenuItem();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TMenuItem (see page 1363) reference is not nil, and only then calls Destroy.

See Also

System::TObject::Free

34.1.5.4 TMenuItem Methods

The methods of the TMenuItem class are listed here.

Protected Methods

	Name	Description
	ActionChange (see page 1369)	This is ActionChange, a member of class TMenuItem.
V	AdvancedDrawItem (see page 1374)	This is AdvancedDrawItem, a member of class TMenuItem.
V	AssignTo (see page 1374)	Copies the properties of an object to a destination object.
V	DefineProperties (see page 1375)	Designates methods for storing an object's unpublished data on a stream such as a form file.
	DoDrawText (see page 1376)	This is DoDrawText, a member of class TMenuItem.
V	DrawItem (see page 1377)	This is DrawItem, a member of class TMenuItem.
	GetActionLinkClass (see page 1377)	This is GetActionLinkClass, a member of class TMenuItem.
	GetAutoHotkeys (see page 1378)	This is GetAutoHotkeys, a member of class TMenuItem.
	GetAutoLineReduction (see page 1378)	This is GetAutoLineReduction, a member of class TMenuItem.
	GetCount (see page 1378)	This is GetCount, a member of class TMenuItem.
	GetHandle (see page 1379)	This is GetHandle, a member of class TMenuItem.
	GetItem (see page 1380)	This is GetItem, a member of class TMenuItem.
	GetMenuIndex (see page 1380)	This is GetMenuIndex, a member of class TMenuItem.
	InsertNewLine (see page 1383)	This is InsertNewLine, a member of class TMenuItem.
V	Loaded (see page 1385)	Initializes the component after the form file has been read into memory.
V	MeasureItem (see page 1386)	This is MeasureItem, a member of class TMenuItem.
V	MenuChanged (see page 1386)	This is MenuChanged, a member of class TMenuItem.
V	Notification (see page 1387)	Forwards notification messages to all owned components.
	SetBreak (see page 1389)	This is SetBreak, a member of class TMenuItem.
	SetCaption (see page 1389)	This is SetCaption, a member of class TMenuItem.
	SetChecked (see page 1390)	This is SetChecked, a member of class TMenuItem.
V	SetChildOrder (see page 1390)	Sets the visual ordering of a child control.
	SetDefault (see page 1390)	This is SetDefault, a member of class TMenuItem.
	SetEnabled (see page 1390)	This is SetEnabled, a member of class TMenuItem.
	SetGroupIndex (see page 1391)	This is SetGroupIndex, a member of class TMenuItem.
	SetImageIndex (see page 1391)	This is SetImageIndex, a member of class TMenuItem.
	SetMenuIndex (see page 1391)	This is SetMenuIndex, a member of class TMenuItem.
	SetRadioItem (see page 1392)	This is SetRadioItem, a member of class TMenuItem.
	SetShortCut (see page 1392)	This is SetShortCut, a member of class TMenuItem.
	SetVisible (see page 1392)	This is SetVisible, a member of class TMenuItem.
	UpdateItems (see page 1392)	This is UpdateItems, a member of class TMenuItem.

Public Methods

	Name	Description
	Add (see page 1369)	This is the overview for the Add method overload.
	Clear (see page 1375)	Removes and frees all menu items listed in the Items (see page 1411) property.
V	Click (see page 1375)	Simulates a mouse click.
	Delete (see page 1376)	Removes a menu item from the Items (see page 1411) property array.
	Find (see page 1377)	Locates a menu item in the Items (see page 1411) property array given its caption.
V	GetChildren (see page 1378)	Enumerates all child components.
	GetEnumerator (see page 1379)	Returns a TComponent enumerator.
	GetImageList (see page 1379)	Returns the Image list that supplies an image for the menu item.
V	GetParentComponent (see page 1380)	Returns the parent component of the menu item.
	GetParentMenu (see page 1380)	Returns the main menu of which the menu item is part.

	HasParent (see page 1381)	Indicates that the menu item has a parent that is responsible for loading and saving its data.
	IndexOf (see page 1381)	Returns the position of a menu item within the Items (see page 1411) array.
	InitiateAction (see page 1382)	Calls the action link's Update method if the menu item is associated with an action link.
	Insert (see page 1382)	Inserts a menu item into a specified position in the Items (see page 1411) array.
	InsertNewLineAfter (see page 1383)	Inserts a separator bar after a specified item from the Items (see page 1411) property array.
	InsertNewLineBefore (see page 1384)	Inserts a separator bar before a specified item from the Items (see page 1411) property array.
	IsLine (see page 1385)	Indicates whether the menu item represents a separator bar.
	NewBottomLine (see page 1386)	Inserts a separator bar at the end of the Items (see page 1411) property array.
	NewTopLine (see page 1387)	Inserts a separator bar at the beginning of the Items (see page 1411) property array.
	Remove (see page 1388)	Removes a menu item from the Items (see page 1411) property array
	RethinkHotkeys (see page 1388)	Adjusts the captions of subitems so that every item has an accelerator and there are no duplicate accelerator keys.
	RethinkLines (see page 1389)	Removes superfluous separator lines from the Items (see page 1411) property array.
	SetParentComponent (see page 1391)	Sets the parent component.

34.1.5.4.1 TMenuItem.ActionChange Method

Pascal

```
procedure ActionChange(Sender: TObject; CheckDefaults: Boolean); dynamic;
```

C++

```
__fastcall ActionChange(TObject * Sender, Boolean CheckDefaults);
```

Description

This is ActionChange, a member of class TMenuItem.

34.1.5.4.2 Add Method

This is the overview for the Add method overload.

Overload List

	Name	Description
	TMenuItem.Add (array of TMenuItem) (see page 1369)	Adds one or more menu items to the end of the Items (see page 1411) property array.
	TMenuItem.Add (TMenuItem) (see page 1369)	Adds one or more menu items to the end of the Items (see page 1411) property array.

34.1.5.4.2.1 TMenuItem.Add Method (TMenuItem)

Adds one or more menu items to the end of the Items (see page 1411) property array.

Pascal

```
procedure Add(Item: TMenuItem); overload;
procedure Add(const AItems: array of TMenuItem); overload;
```

C++

```
__fastcall Add(TMenuItem Item);
__fastcall Add(const array of TMenuItem AItems);
```

Description

Use Add to add new menu items to the dropdown menu for this menu item. If this menu item is the value of the Items (see page 1411) property of a main menu or pop-up menu, Add adds the items to the end of the menu.

Specify a single new menu item to add as the value of the Item parameter, or an array of new menu items as the value of the AItems parameter. AItems_Size is the index of the last item in the AItems array (one less than the number of items).

C++ Examples:

```
/*
This example assumes that the main form of the application
has a main menu with a menu item. The following code adds a
separator, and the name of all forms to the menu. Do not
access FormCount in the mainform's FormCreate.  The other
forms do not exist yet!
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMenuItem *NewItem;
  // first create the separator
  NewItem = new TMenuItem(this); // If you change the AOwner (this) to MainMenu1->Items[1] and
then you add the new menu you will get "Menu inserted twice"?  Doesn't happen in Delphi.
  NewItem->Name = "Separator";   // The owner will cleanup these menu items.
  NewItem->Caption = "-";
  // add the new item to the Windows menu
  MainMenu1->Items->Items[1]->Add(NewItem);
  // now create and add a menu item for each form
  for  (int I = 0; I < Screen->FormCount; I++)
  {
    NewItem = new TMenuItem(this); // The owner will cleanup these menu items.
    NewItem->Caption = Screen->Forms[I]->Name + "Item";
    NewItem->Name = Screen->Forms[I]->Name;
    MainMenu1->Items->Items[1]->Add(NewItem);
  }
}
/*
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
    {
        switch (ClickedItem->Tag)
        {
```

```
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
            }
        }
    }
}

/*
To edit or destroy an item, grab its pointer via the Items
property.
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}

void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
This example assumes that the main form of the application
has a main menu with a menu item. The following code adds a
separator, and the name of all forms to the menu. Do not
access FormCount in the mainform's FormCreate.  The other
forms do not exist yet!
}
procedure TForm1.Button1Click(Sender: TObject);
```

```
  var
    NewItem: TMenuItem;
    I : integer;
  begin
    { first create the separator }
    NewItem := TMenuItem.Create(MainMenu1.Items[1]);
    NewItem.Name:= 'Separator';
    NewItem.Caption := '-';
    { add the new item to the Windows menu }
    MainMenu1.Items[1].Add(NewItem);
    { now create and add a menu item for each form }
    for  I := 0 to Screen.FormCount-1 do
    begin
      NewItem := TMenuItem.Create(MainMenu1.Items[1]);
      NewItem.Caption := Screen.Forms[I].Name + 'Item';
      NewItem.Name := Screen.Forms[I].Name;
      MainMenu1.Items[1].Add(NewItem);
     end;
  end;

  procedure TForm1.FormDestroy(Sender: TObject);
  var
    I : Integer;
  begin
    // Count down, not up!
    for  I := MainMenu1.Items[1].Count - 1 downto 0 do
    begin
      MainMenu1.Items[1].Items[I].Free;
    end;
  end;
  {
  This example demonstrates how to add and delete menu items
  to a popup menu at runtime and assign an event handler to
  the OnClick event.  Place a TPopupMenu and three buttons on
  the form named "AddButton", "EditButton",  and
  "DestroyButton" and add OnClick events to all three buttons.
  Put the TPopupMenu in the PopupMenu property of the form.
  Place the PopupMenuItemsClick procedure in the TForm1 type
  declaration so that it can be used as the method call for
  the menu item OnClick event.
  }

  type
    TForm1 = class(TForm)
      AddButton: TButton;
      EditButton: TButton;
      DestroyButton: TButton;
      PopupMenu1: TPopupMenu;
      procedure AddButtonClick(Sender: TObject);
      procedure EditButtonClick(Sender: TObject);
      procedure DestroyButtonClick(Sender: TObject);
      procedure FormCreate(Sender: TObject);
    private
      { Private declarations }
    public
      procedure PopupMenuItemsClick(Sender: TObject);

    end;

  var
    Form1: TForm1;

  implementation

  {$R *.dfm}
```

```
procedure TForm1.AddButtonClick(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
    // The owner (PopupMenu1) will cleanup this menu item.
  NewItem := TMenuItem.Create(PopupMenu1); // create the new item
  index := PopupMenu1.Items.Count;
  PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
  NewItem.Caption := 'Menu Item ' + IntToStr(index);
  NewItem.Tag := index;
  NewItem.OnClick :=
    PopupMenuItemsClick; // assign it an event handler
end;

procedure TForm1.PopupMenuItemsClick(Sender: TObject);
begin
  with Sender as TMenuItem do
  begin
    case Tag of
      0:  ShowMessage('first item clicked');
      1:  ShowMessage('second item clicked');
      2:  ShowMessage('third item clicked');
      3:  ShowMessage('fourth item clicked');
    end;
  end;
end;

{
To edit or destroy an item, grab its pointer via the Items
property.
}
procedure TForm1.EditButtonClick(Sender: TObject);
var
  ItemToEdit: TMenuItem;
begin
  ItemToEdit := PopupMenu.Items[1];
  ItemToEdit.Caption := 'Changed Caption';
end;

procedure TForm1.DestroyButtonClick(Sender: TObject);
var
  ItemToDelete: TMenuItem;
begin
  ItemToDelete := PopupMenu.Items[2];
  ItemToDelete.Free;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
  for index := 0 to 3 do
  begin
      // The owner (PopupMenu1) will cleanup this menu item.
    NewItem := TMenuItem.Create(PopupMenu1); // create the new item
    PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
    NewItem.Caption := 'Menu Item ' + IntToStr(index);
    NewItem.Tag := index;
    NewItem.OnClick :=
      PopupMenuItemsClick; // assign it an event handler
  end;
end;
```

See Also

Delete (see page 1376)

Insert (see page 1382)

Items (see page 1411)

Remove (see page 1388)

Items (see page 1346)

NewItem (see page 1448)

NewSubMenu (see page 1452)

34.1.5.4.3 TMenuItem.AdvancedDrawItem Method

Pascal

```
procedure AdvancedDrawItem(ACanvas: TCanvas; ARect: TRect; State: TOwnerDrawState; TopLevel:
Boolean); virtual;
```

C++

```
virtual __fastcall AdvancedDrawItem(TCanvas ACanvas, TRect ARect, TOwnerDrawState State,
Boolean TopLevel);
```

Description

This is AdvancedDrawItem, a member of class TMenuItem.

34.1.5.4.4 TMenuItem.AssignTo Method

Copies the properties of an object to a destination object.

Pascal

```
procedure AssignTo(Dest: TPersistent); override;
```

C++

```
virtual __fastcall AssignTo(TPersistent * Dest);
```

Description

Override the AssignTo method to extend the functionality of the Assign method of destination objects so that they handle newly created object classes. When defining a new object class, override the Assign method for every existing object class that should be able to copy its properties to the new class. Override the AssignTo method for every existing class to which the new class can copy.

The Assign method of TPersistent calls AssignTo if the descendant object does not succeed in copying the properties of a source object. The AssignTo method defined by TPersistent raises an EConvertError exception.

For example, given the following code in which A and B are instance variables:

```
A.Assign(B); {Delphi}
A->Assign(B); // C++
```

if A knows how to handle B, then it does so and returns. If A doesn't know how to handle B's type, execution will trickle to the TPersistent version of Assign, which calls:

```
B.AssignTo(A); {Delphi}
B->AssignTo(A); // C++
```

If B knows how to copy to A, the assignment succeeds. Otherwise, TPersistent raises an exception.

See Also

Assign

EConvertError

34.1.5.4.5 TMenuItem.Clear Method

Removes and frees all menu items listed in the Items (see page 1411) property.

Pascal

```
procedure Clear;
```

C++

```
__fastcall Clear();
```

Description

Use Clear to free all the items listed in the Items (see page 1411) property array.

See Also

Items (see page 1411)

Delete (see page 1376)

34.1.5.4.6 TMenuItem.Click Method

Simulates a mouse click.

Pascal

```
procedure Click; virtual;
```

C++

```
virtual __fastcall Click();
```

Description

Click generates an OnClick (see page 1417) event, as if the user had clicked the menu item. Unlike a real mouse click, Click does not generate a WM_COMMAND (see page 1684) message to the window associated with the menu.

Component writers can override click to provide a different response when the user selects the menu item.

See Also

Command (see page 1401)

OnClick (see page 1417)

34.1.5.4.7 TMenuItem.DefineProperties Method

Designates methods for storing an object's unpublished data on a stream such as a form file.

Pascal

```
procedure DefineProperties(Filer: TFiler); override;
```

C++

```
virtual __fastcall DefineProperties(TFiler Filer);
```

Description

TComponent overrides the DefineProperties method defined in TPersistent to define "fake" Top and Left properties. These are defined so that components that are not controls can be manipulated at design-time. However, the Top and Left properties are hidden, that is, they are not published, because only controls appear at run-time.

DefineProperties is virtual; descendant classes can override it. When overriding DefineProperties, be aware that the Ancestor property of Filer might be set, and that this property can determine whether or not it is appropriate to write properties.

DefineProperties is called automatically as part of the component streaming system; do not call it directly.

See Also

Ancestor

DefineBinaryProperty

DefineProperties

DefineProperty

34.1.5.4.8 TMenuItem.Delete Method

Removes a menu item from the Items (see page 1411) property array.

Pascal

```
procedure Delete(Index: Integer);
```

C++

```
__fastcall Delete(int Index);
```

Description

Call Delete to remove a menu item and all its submenus from a menu. The Index parameter is the index of the item to be deleted, where 0 specifies the first item in the menu, 1 specifies the second item, and so on.

Delete does not free the removed item's memory. To remove an item from the menu and free its memory, call the Free method in the item you want to remove.

To temporarily hide a menu item, without deleting it, use the Visible (see page 1416) property.

See Also

Add (see page 1369)

Insert (see page 1382)

Items (see page 1411)

Remove (see page 1388)

Visible (see page 1416)

System::TObject::Free

34.1.5.4.9 TMenuItem.DoDrawText Method

Pascal

```
procedure DoDrawText(ACanvas: TCanvas; const ACaption: string; var Rect: TRect; Selected:
Boolean; Flags: Longint);
```

C++

```
__fastcall DoDrawText(TCanvas ACanvas, const UnicodeString ACaption, TRect Rect, Boolean
Selected, Longint Flags);
```

Description

This is DoDrawText, a member of class TMenuItem.

34.1.5.4.10 TMenuItem.DrawItem Method

Pascal

```
procedure DrawItem(ACanvas: TCanvas; ARect: TRect; Selected: Boolean); virtual;
```

C++

```
virtual __fastcall DrawItem(TCanvas ACanvas, TRect ARect, Boolean Selected);
```

Description

This is DrawItem, a member of class TMenuItem.

34.1.5.4.11 TMenuItem.Find Method

Locates a menu item in the Items (see page 1411) property array given its caption.

Pascal

```
function Find(ACaption: string): TMenuItem;
```

C++

```
__fastcall TMenuItem Find(UnicodeString ACaption);
```

Description

Use Find to locate the menu item in the Items (see page 1411) property array with the caption specified by the ACaption parameter. Find ignores accelerator characters when comparing item captions to the ACaption parameter.

Find returns the first menu item in the Items (see page 1411) property array with the specified caption. If no item is found, Find returns nil (Delphi) or NULL (C++).

See Also

Items (see page 1411)

Caption (see page 1396)

34.1.5.4.12 TMenuItem.GetActionLinkClass Method

Pascal

```
function GetActionLinkClass: TMenuActionLinkClass; dynamic;
```

C++

```
__fastcall TMenuActionLinkClass GetActionLinkClass();
```

Description

This is GetActionLinkClass, a member of class TMenuItem.

34.1.5.4.13 TMenuItem.GetAutoHotkeys Method

Pascal

```
function GetAutoHotkeys: Boolean;
```

C++

```
__fastcall Boolean GetAutoHotkeys();
```

Description

This is GetAutoHotkeys, a member of class TMenuItem.

34.1.5.4.14 TMenuItem.GetAutoLineReduction Method

Pascal

```
function GetAutoLineReduction: Boolean;
```

C++

```
__fastcall Boolean GetAutoLineReduction();
```

Description

This is GetAutoLineReduction, a member of class TMenuItem.

34.1.5.4.15 TMenuItem.GetChildren Method

Enumerates all child components.

Pascal

```
procedure GetChildren(Proc: TGetChildProc; Root: TComponent); override;
```

C++

```
virtual __fastcall GetChildren(TGetChildProc Proc, TComponent * Root);
```

Description

GetChildren is used internally in the component streaming system. It is not necessary to call it directly.

GetChildren expects a TGetChildProc routine that receives all enumerated components.

`Root` represents the owner of the components that will be enumerated.

This method does nothing in TComponent and is expected to be overridden in descendants.

See Also

TGetChildProc

GetParentComponent

34.1.5.4.16 TMenuItem.GetCount Method

Pascal

```
function GetCount: Integer;
```

C++

```
__fastcall int GetCount();
```

Description

This is GetCount, a member of class TMenuItem.

34.1.5.4.17 TMenuItem.GetEnumerator Method

Returns a TComponent enumerator.

Pascal

```
function GetEnumerator: TMenuItemEnumerator;
```

C++

```
__fastcall TMenuItemEnumerator GetEnumerator();
```

Description

GetEnumerator returns a TComponentEnumerator reference, which enumerates the components contained within a specified containing component.

To process all these sub components, call the TComponentEnumerator GetCurrent method within a While MoveNext do loop.

See Also

TComponentEnumerator

34.1.5.4.18 TMenuItem.GetHandle Method

Pascal

```
function GetHandle: HMENU;
```

C++

```
__fastcall HMENU GetHandle();
```

Description

This is GetHandle, a member of class TMenuItem.

34.1.5.4.19 TMenuItem.GetImageList Method

Returns the Image list that supplies an image for the menu item.

Pascal

```
function GetImageList: TCustomImageList;
```

C++

```
__fastcall TCustomImageList GetImageList();
```

Description

Use GetImageList to access the list of images that contains the image for this menu item. The ImageIndex (see page 1411) property identifies a specific image from the list that GetImageList returns.

GetImageList first checks the Parent (see page 1414) of this menu item for a SubMenuImages (see page 1415) property. If the parent menu item does not maintain a list of images, it checks that menu item's parent, and so on until reaching a top-level menu item (a menu item where Parent (see page 1414) is not set). If none of the parent menu items has a SubMenuImages (see page 1415) property, GetImageList checks the top-level menu for an Images property.

GetImageList returns the first image list it finds. If it doesn't find an image list, GetImageList returns nil (Delphi) or NULL (C++).

See Also

ImageIndex (see page 1411)

SubMenuImages (see page 1415)

Images (see page 1345)

Parent (see page 1414)

34.1.5.4.20 TMenuItem.GetItem Method

Pascal

```
function GetItem(Index: Integer): TMenuItem;
```

C++

```
__fastcall TMenuItem GetItem(int Index);
```

Description

This is GetItem, a member of class TMenuItem.

34.1.5.4.21 TMenuItem.GetMenuIndex Method

Pascal

```
function GetMenuIndex: Integer;
```

C++

```
__fastcall int GetMenuIndex();
```

Description

This is GetMenuIndex, a member of class TMenuItem.

34.1.5.4.22 TMenuItem.GetParentComponent Method

Returns the parent component of the menu item.

Pascal

```
function GetParentComponent: TComponent; override;
```

C++

```
virtual __fastcall TComponent * GetParentComponent();
```

Description

GetParentComponent is called by the streaming system that loads and saves components. It ensures that objects are loaded and saved with their parent objects. GetParentComponent is overridden in descendant classes to return the appropriate parent for their particular type of component. TMenuItem (see page 1363) overrides it to identify its associated menu, if it has one, as its parent.

34.1.5.4.23 TMenuItem.GetParentMenu Method

Returns the main menu of which the menu item is part.

Pascal

```
function GetParentMenu: TMenu;
```

C++

```
__fastcall TMenu GetParentMenu();
```

Description

GetParentMenu returns the menu that contains the menu item.

See Also

System::TObject::Free

34.1.5.4.24 TMenuItem.HasParent Method

Indicates that the menu item has a parent that is responsible for loading and saving its data.

Pascal

```
function HasParent: Boolean; override;
```

C++

```
virtual __fastcall Boolean HasParent();
```

Description

TMenuItem (see page 1363) overrides HasParent to return true, indicating that menu items have a parent. Applications seldom call the HasParent method. It is used by the streaming system that loads and saves forms to determine when another object is responsible for writing a component to a stream.

See Also

Parent (see page 1414)

HasParent

34.1.5.4.25 TMenuItem.IndexOf Method

Returns the position of a menu item within the Items (see page 1411) array.

Pascal

```
function IndexOf(Item: TMenuItem): Integer;
```

C++

```
__fastcall int IndexOf(TMenuItem Item);
```

Description

Call IndexOf to locate a menu item in the submenu headed by this menu item. IndexOf returns 0 for the first menu item, 1 for the second menu item, and so on. If the menu item is not in the submenu, IndexOf returns -1.

Note: IndexOf may not correspond to the position of the item in the menu the user sees if the menu includes some items that are not visible.

See Also

Items (see page 1411)

MenuIndex (see page 1413)

Visible (see page 1416)

34.1.5.4.26 TMenuItem.InitiateAction Method

Calls the action link's Update method if the menu item is associated with an action link.

Pascal

```
procedure InitiateAction; virtual;
```

C++

```
virtual __fastcall InitiateAction();
```

Description

InitiateAction calls the action link, if there is one for the menu item, to update the association between the action and the menu item.

See Also

Items (see page 1411)

MenuIndex (see page 1413)

Visible (see page 1416)

34.1.5.4.27 TMenuItem.Insert Method

Inserts a menu item into a specified position in the Items (see page 1411) array.

Pascal

```
procedure Insert(Index: Integer; Item: TMenuItem);
```

C++

```
__fastcall Insert(int Index, TMenuItem Item);
```

Description

Use Insert to insert a new menu item to the dropdown menu for this menu item. If this menu item is the value of the Items (see page 1411) property of a main menu or pop-up menu, Insert inserts an item into that menu. Specify the new menu item to add as the value of the Item parameter, and the position in the Items (see page 1411) array as the value of the Index parameter.

C++ Examples:

```
/*
This code inserts a menu item after the first item in a menu
named FileMenu:
*/
void __fastcall TForm1::InsertMenuItemClick(TObject *Sender)
{
  TMenuItem *NewItem = new TMenuItem(FileMenu);
  try
  {
    NewItem->Caption = "Form1";
    FileMenu->Insert(1, NewItem);
  }
  catch (Exception &e)
  {
    delete NewItem;
    throw;
  }
}
```

Delphi Examples:

```
{
This code inserts a menu item after the first item in a menu
named FileMenu:
}
procedure TForm1.InsertMenuItemClick(Sender: TObject);
var
  NewItem: TMenuItem;
begin
  NewItem := TMenuItem.Create(FileMenu);
  try
    NewItem.Caption := 'Do this';
    FileMenu.Insert(1, NewItem);
  except
    NewItem.Free; // Throw away the menu item if there is an error.
    raise; // reraise the exception
  end;
  // Don't free the menu item on success.  Need it for the menu!
end;

// Free all the menu items that we have created dynamically.
procedure TForm1.FormDestroy(Sender: TObject);
var
  I : Integer;
begin
  for I := FileMenu.Count - 1 downto 0 do  // Index backwards because Count changes as you
remove items.
    if FileMenu.Items[I].Name = '' then  // Items created dynamically have not name.
      FileMenu.Items[I].Free;
end;
```

See Also

Add (see page 1369)

Delete (see page 1376)

Items (see page 1411)

Remove (see page 1388)

Items (see page 1346)

34.1.5.4.28 TMenuItem.InsertNewLine Method

Pascal

```
function InsertNewLine(ABefore: Boolean; AItem: TMenuItem): Integer;
```

C++

```
__fastcall int InsertNewLine(Boolean ABefore, TMenuItem AItem);
```

Description

This is InsertNewLine, a member of class TMenuItem.

34.1.5.4.29 TMenuItem.InsertNewLineAfter Method

Inserts a separator bar after a specified item from the Items (see page 1411) property array.

Pascal

```
function InsertNewLineAfter(AItem: TMenuItem): Integer;
```

C++

```
__fastcall int InsertNewLineAfter(TMenuItem AItem);
```

Description

Use InsertNewLineAfter to insert a separator bar after the menu item specified by AItem. If AItem is not an item in the Items (see page 1411) property array, InsertNewLineAfter raises an exception.

InsertNewLineAfter returns the index of the newly inserted separator item in the Items (see page 1411) property array.

Tip: When inserting separator bars at runtime, it is a good idea to set AutoLineReduction (see page 1395) to true so that superfluous separators can be automatically removed.

See Also

Insert (see page 1382)

InsertNewLineBefore (see page 1384)

Items (see page 1411)

NewBottomLine (see page 1386)

NewTopLine (see page 1387)

AutoLineReduction (see page 1395)

34.1.5.4.30 TMenuItem.InsertNewLineBefore Method

Inserts a separator bar before a specified item from the Items (see page 1411) property array.

Pascal

```
function InsertNewLineBefore(AItem: TMenuItem): Integer;
```

C++

```
__fastcall int InsertNewLineBefore(TMenuItem AItem);
```

Description

Use InsertNewLineBefore to insert a separator bar before the menu item specified by AItem. If AItem is not an item in the Items (see page 1411) property array, InsertNewLineBefore raises an exception.

InsertNewLineBefore returns the index of the newly inserted separator item in the Items (see page 1411) property array.

Tip: When inserting separator bars at runtime, it is a good idea to set AutoLineReduction (see page 1395) to true so that superfluous separators can be automatically removed.

See Also

Insert (see page 1382)

InsertNewLineAfter (see page 1383)

Items (see page 1411)

NewBottomLine (see page 1386)

NewTopLine (see page 1387)

AutoLineReduction (see page 1395)

34.1.5.4.31 TMenuItem.IsLine Method

Indicates whether the menu item represents a separator bar.

Pascal

```
function IsLine: Boolean;
```

C++

```
__fastcall Boolean IsLine();
```

Description

Call IsLine to determine whether the menu item is a separator bar. IsLine checks the Caption (see page 1396) of the menu item and returns true if the value of Caption (see page 1396) is '-'. Menu items with this Caption (see page 1396) appear as separator bars.

See Also

Caption (see page 1396)

34.1.5.4.32 TMenuItem.Loaded Method

Initializes the component after the form file has been read into memory.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

Do not call the protected Loaded method. The streaming system calls this method after it loads the component's form from a stream.

When the streaming system loads a form or data module from its form file, it first constructs the form component by calling its constructor, then reads its property values from the form file. After reading all the property values for all the components, the streaming system calls the Loaded methods of each component in the order the components were created. This gives the components a chance to initialize any data that depends on the values of other components or other parts of itself.

Note: All references to sibling components are resolved by the time Loaded is called. Loaded is the first place that sibling pointers can be used after being streamed in.

As implemented in TComponent, Loaded clears the csLoading flag in the ComponentState property, indicating that the component is no longer loading.

Warning: Loaded may be called multiple times on inherited forms. It is called every time a level of inheritance is streamed in. Do not allocate memory in an overridden Loaded method without first checking that the memory has not been allocated in a previous call.

See Also

ComponentState

Create

TReader

34.1.5.4.33 TMenuItem.MeasureItem Method

Pascal

```
procedure MeasureItem(ACanvas: TCanvas; var Width: Integer; var Height: Integer); virtual;
```

C++

```
virtual __fastcall MeasureItem(TCanvas ACanvas, int Width, int Height);
```

Description

This is MeasureItem, a member of class TMenuItem.

34.1.5.4.34 TMenuItem.MenuChanged Method

Pascal

```
procedure MenuChanged(Rebuild: Boolean); virtual;
```

C++

```
virtual __fastcall MenuChanged(Boolean Rebuild);
```

Description

This is MenuChanged, a member of class TMenuItem.

34.1.5.4.35 TMenuItem.NewBottomLine Method

Inserts a separator bar at the end of the Items (see page 1411) property array.

Pascal

```
function NewBottomLine: Integer;
```

C++

```
__fastcall int NewBottomLine();
```

Description

Use NewBottomLine to insert a separator bar at the end of the items property array. This is useful when building up submenus dynamically at runtime.

NewBottomLine returns the index of the newly inserted separator item in the Items (see page 1411) property array.

Tip: When inserting separator bars at runtime, it is a good idea to set AutoLineReduction (see page 1395) to true so that superfluous separators can be automatically removed.

See Also

Add (see page 1369)

InsertNewLineBefore (see page 1384)

Items (see page 1411)

InsertNewLineAfter (see page 1383)

NewTopLine (see page 1387)

AutoLineReduction (see page 1395)

34.1.5.4.36 TMenuItem.NewTopLine Method

Inserts a separator bar at the beginning of the Items (see page 1411) property array.

Pascal

```
function NewTopLine: Integer;
```

C++

```
__fastcall int NewTopLine();
```

Description

Use NewTopLine to insert a separator bar at the beginning of the items property array. This is useful when building up submenus dynamically at runtime.

NewTopLine returns the index of the newly inserted separator item in the Items (see page 1411) property array.

Tip: When inserting separator bars at runtime, it is a good idea to set AutoLineReduction (see page 1395) to true so that superfluous separators can be automatically removed.

See Also

Add (see page 1369)

InsertNewLineBefore (see page 1384)

Items (see page 1411)

InsertNewLineAfter (see page 1383)

NewBottomLine (see page 1386)

AutoLineReduction (see page 1395)

34.1.5.4.37 TMenuItem.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

34.1.5.4.38 TMenuItem.Remove Method

Removes a menu item from the Items (see page 1411) property array

Pascal

```
procedure Remove(Item: TMenuItem);
```

C++

```
__fastcall Remove(TMenuItem Item);
```

Description

Call Remove to remove a menu item and all its submenus from a menu. The Item parameter is the item to be removed. If Item is not a menu item in the Items (see page 1411) array, Remove raises an exception.

Remove does not free the item's memory. To remove an item from the menu and free its memory, call the Free method in the item you want to remove.

To temporarily hide a menu item, without deleting it, use the Visible (see page 1416) property.

See Also

Add (see page 1369)

Delete (see page 1376)

EMenuError (see page 1321)

Insert (see page 1382)

Items (see page 1411)

Visible (see page 1416)

34.1.5.4.39 TMenuItem.RethinkHotkeys Method

Adjusts the captions of subitems so that every item has an accelerator and there are no duplicate accelerator keys.

Pascal

```
function RethinkHotkeys: Boolean;
```

C++

```
__fastcall Boolean RethinkHotkeys();
```

Description

Call RethinkHotkeys to adjust the accelerator keys for the items in the Items (see page 1411) property and in any submenu of those items. RethinkHotkeys checks for items that do not have an accelerator key or that have duplicate accelerator keys. It then replaces any duplicate accelerators with unique characters and adds accelerator keys to items that don't have them. RethinkHotkeys gives first priority to existing accelerator key mappings, but may change them if necessary to ensure uniqueness.

RethinkHotkeys lets a menu adjust the accelerator keys when the AutoHotkeys (see page 1394) property is false. For any menu or submenu that has AutoHotkeys (see page 1394) set to true, this adjustment occurs automatically.

RethinkHotkeys returns true if RethinkHotkeys changed the Caption (see page 1396) property of any menu items, false otherwise.

See Also

AutoHotkeys (see page 1394)

AutoHotkeys (see page 1343)

Caption (see page 1396)

Items (see page 1411)

34.1.5.4.40 TMenuItem.RethinkLines Method

Removes superfluous separator lines from the Items (see page 1411) property array.

Pascal

```
function RethinkLines: Boolean;
```

C++

```
__fastcall Boolean RethinkLines();
```

Description

Call RethinkLines to remove superfluous separators from the items in the Items (see page 1411) property and in any submenu of those items. RethinkLines removes separators that appear at the very beginning or end of a submenu and separators that immediately follow another separator.

RethinkLines lets a menu remove superfluous separators when the AutoLineReduction (see page 1395) property is false. For any menu or submenu that has AutoLineReduction (see page 1395) set to true, this adjustment occurs automatically.

RethinkLines returns true if RethinkLines removed any menu items, false otherwise.

See Also

AutoLineReduction (see page 1395)

AutoLineReduction (see page 1344)

IsLine (see page 1385)

Items (see page 1411)

34.1.5.4.41 TMenuItem.SetBreak Method

Pascal

```
procedure SetBreak(Value: TMenuBreak);
```

C++

```
__fastcall SetBreak(TMenuBreak Value);
```

Description

This is SetBreak, a member of class TMenuItem.

34.1.5.4.42 TMenuItem.SetCaption Method

Pascal

```
procedure SetCaption(const Value: string);
```

C++

```
__fastcall SetCaption(const UnicodeString Value);
```

Description

This is SetCaption, a member of class TMenuItem.

34.1.5.4.43 TMenuItem.SetChecked Method

Pascal

```
procedure SetChecked(Value: Boolean);
```

C++

```
__fastcall SetChecked(Boolean Value);
```

Description

This is SetChecked, a member of class TMenuItem.

34.1.5.4.44 TMenuItem.SetChildOrder Method

Sets the visual ordering of a child control.

Pascal

```
procedure SetChildOrder(Child: TComponent; Order: Integer); override;
```

C++

```
virtual __fastcall SetChildOrder(TComponent * Child, int Order);
```

Description

SetChildOrder is used internally in the component streaming system. It is not necessary to call it directly.

Derived classes override this method to actually implement the ordering. TWinControl uses this ordering information to Z-align the controls.

See Also

SetZOrder

34.1.5.4.45 TMenuItem.SetDefault Method

Pascal

```
procedure SetDefault(Value: Boolean);
```

C++

```
__fastcall SetDefault(Boolean Value);
```

Description

This is SetDefault, a member of class TMenuItem.

34.1.5.4.46 TMenuItem.SetEnabled Method

Pascal

```
procedure SetEnabled(Value: Boolean);
```

C++

```
__fastcall SetEnabled(Boolean Value);
```

Description

This is SetEnabled, a member of class TMenuItem.

34.1.5.4.47 TMenuItem.SetGroupIndex Method

Pascal

```
procedure SetGroupIndex(Value: Byte);
```

C++

```
__fastcall SetGroupIndex(Byte Value);
```

Description

This is SetGroupIndex, a member of class TMenuItem.

34.1.5.4.48 TMenuItem.SetImageIndex Method

Pascal

```
procedure SetImageIndex(Value: TImageIndex);
```

C++

```
__fastcall SetImageIndex(TImageIndex Value);
```

Description

This is SetImageIndex, a member of class TMenuItem.

34.1.5.4.49 TMenuItem.SetMenuIndex Method

Pascal

```
procedure SetMenuIndex(Value: Integer);
```

C++

```
__fastcall SetMenuIndex(int Value);
```

Description

This is SetMenuIndex, a member of class TMenuItem.

34.1.5.4.50 TMenuItem.SetParentComponent Method

Sets the parent component.

Pascal

```
procedure SetParentComponent(Value: TComponent); override;
```

C++

```
virtual __fastcall SetParentComponent(TComponent * Value);
```

Description

SetParentComponent is used internally in the component streaming system. It is not necessary to call it directly.

Derived classes override this method to implement proper handling for parenting.

See Also

SetParent

Parent

34.1.5.4.51 TMenuItem.SetRadioItem Method

Pascal

```
procedure SetRadioItem(Value: Boolean);
```

C++

```
__fastcall SetRadioItem(Boolean Value);
```

Description

This is SetRadioItem, a member of class TMenuItem.

34.1.5.4.52 TMenuItem.SetShortCut Method

Pascal

```
procedure SetShortCut(Value: TShortCut);
```

C++

```
__fastcall SetShortCut(TShortCut Value);
```

Description

This is SetShortCut, a member of class TMenuItem.

34.1.5.4.53 TMenuItem.SetVisible Method

Pascal

```
procedure SetVisible(Value: Boolean);
```

C++

```
__fastcall SetVisible(Boolean Value);
```

Description

This is SetVisible, a member of class TMenuItem.

34.1.5.4.54 TMenuItem.UpdateItems Method

Pascal

```
procedure UpdateItems;
```

C++

```
__fastcall UpdateItems();
```

Description

This is UpdateItems, a member of class TMenuItem.

34.1.5.5 TMenuItem Properties

The properties of the TMenuItem class are listed here.

Protected Properties

	Name	Description
	ActionLink (see page 1394)	This is ActionLink, a member of class TMenuItem.

Public Properties

	Name	Description
R	Command (see page 1401)	Specifies the Windows Command ID associated with the menu item.
R	Count (see page 1402)	Indicates the number of subitems of the menu item.
R	Handle (see page 1409)	Indicates the Windows menu handle of the drop-down menu associated with the menu item.
R	Items (see page 1411)	Lists the menu items in the submenu of the menu item.
	MenuIndex (see page 1413)	Indicates the index of the menu item within its parent menu.
R	Parent (see page 1414)	Identifies the parent menu item of this menu item.

Published Properties

	Name	Description
	Action (see page 1393)	Designates the action associated with the menu item.
	AutoCheck (see page 1394)	Indicates whether the menu item's checked state toggles automatically when the item is clicked.
	AutoHotkeys (see page 1394)	Determines whether the accelerator keys for submenu items can be reset automatically.
	AutoLineReduction (see page 1395)	Determines whether redundant separator bars are automatically removed from the submenu.
	Bitmap (see page 1395)	Designates a bitmap that is drawn for the menu item.
	Break (see page 1396)	Determines whether the menu item starts a new column in the menu.
	Caption (see page 1396)	Specifies the text of the menu item.
	Checked (see page 1401)	Specifies whether a check mark should appear beside the Caption (see page 1396).
	Default (see page 1406)	Specifies whether the menu item is invoked when the parent item is double clicked.
	Enabled (see page 1406)	Specifies whether the menu item is enabled.
	GroupIndex (see page 1408)	Identifies the logical group to which the menu item belongs.
	HelpContext (see page 1410)	Specifies the help context ID associated with the menu item.
	Hint (see page 1410)	Specifies the text string that can appear when the user moves the mouse pointer over a menu item.
	ImageIndex (see page 1411)	Indicates which image maintained by the parent menu appears next to the menu item.
	RadioItem (see page 1414)	Specifies whether the menu item is mutually exclusive with other menu items in its group.
	ShortCut (see page 1415)	Specifies the key combination users can type to quickly access the menu item.
	SubMenuImages (see page 1415)	Lists the images that can appear beside individual items in a submenu of this menu item.
	Visible (see page 1416)	Specifies whether the menu item appears in the menu.

34.1.5.5.1 TMenuItem.Action Property

Designates the action associated with the menu item.

Pascal

```
property Action: TBasicAction;
```

C++

```
__property TBasicAction Action;
```

Description

Action designates the action object that is associated with the menu item. Actions can be associated with a menu item by using the action list editor at design-time. Actions are a way of centralizing user responses to commands.

34.1.5.5.2 TMenuItem.ActionLink Property

Pascal

```
property ActionLink: TMenuActionLink;
```

C++

```
__property TMenuActionLink ActionLink;
```

Description

This is ActionLink, a member of class TMenuItem.

34.1.5.5.3 TMenuItem.AutoCheck Property

Indicates whether the menu item's checked state toggles automatically when the item is clicked.

Pascal

```
property AutoCheck: Boolean;
```

C++

```
__property Boolean AutoCheck;
```

Description

When AutoCheck is true, then every time the menu item is clicked, the value of the Checked (see page 1401) property toggles automatically before the OnClick (see page 1417) event occurs. When AutoCheck is false, the application must explicitly set the value of the Checked (see page 1401) property (for example, in an OnClick (see page 1417) event handler).

See Also

Checked (see page 1401)

OnClick (see page 1417)

34.1.5.5.4 TMenuItem.AutoHotkeys Property

Determines whether the accelerator keys for submenu items can be reset automatically.

Pascal

```
property AutoHotkeys: TMenuItemAutoFlag;
```

C++

```
__property TMenuItemAutoFlag AutoHotkeys;
```

Description

Set AutoHotkeys to indicate whether the accelerator keys for the items in the Items (see page 1411) property can be reset if necessary to ensure that every menu item has an accelerator key and no two items have the same accelerator key. This ensures that menu items added dynamically at runtime can have accelerator keys that do not introduce conflicts. TMenuItem (see page 1363) tries to preserve all existing accelerator key mappings when adjusting the accelerator keys, but may be forced to change

some in order to ensure that every item has a unique accelerator.

Before the menu is displayed, it checks and adjusts the accelerator key mappings if AutoHotkeys is maAutomatic or if AutoHotkeys is maParent and the parent menu item adjusts accelerator key mappings. Otherwise, the menu does not alter the items listed in the Items (see page 1411) property, but may adjust the accelerator keys for submenus, depending on the AutoHotkeys property of the menu item that is the parent of the submenu.

Note: When AutoHotkeys disallows automatic adjustment of accelerator keys, an application can still adjust the menu items by calling the RethinkHotkeys (see page 1388) method. AutoHotkeys merely controls whether this adjustment occurs automatically.

See Also

AutoHotkeys (see page 1343)

Items (see page 1411)

RethinkHotkeys (see page 1388)

34.1.5.5.5 TMenuItem.AutoLineReduction Property

Determines whether redundant separator bars are automatically removed from the submenu.

Pascal

```
property AutoLineReduction: TMenuItemAutoFlag;
```

C++

```
__property TMenuItemAutoFlag AutoLineReduction;
```

Description

Set AutoLineReduction to indicate whether the menu item should automatically ensure that the submenu defined by its Items (see page 1411) property does not start or end with a separator bar and that two separator bars do not appear next to each other. This ensures that menus built dynamically at runtime do not have separator bars in undesired positions.

Before the menu is displayed, the menu item removes superfluous separator bars if AutoLineReduction is maAutomatic or AutoLineReduction is maParent and the parent menu item removes superfluous separator bars. Otherwise, the menu item does not remove separators from the Items (see page 1411) property, but may remove superfluous separators from submenus, depending on the AutoLineReduction property of the menu item that is the parent of the submenu.

Note: When AutoLineReduction disallows automatic removal of separator bars, an application can still remove superfluous separators by calling the RethinkLines (see page 1389) method. AutoLineReduction merely controls whether this adjustment occurs automatically.

See Also

AutoLineReduction (see page 1344)

Items (see page 1411)

RethinkLines (see page 1389)

34.1.5.5.6 TMenuItem.Bitmap Property

Designates a bitmap that is drawn for the menu item.

Pascal

```
property Bitmap: TBitmap;
```

C++

```
__property TBitmap Bitmap;
```

Description

Use Bitmap to provide a custom image that appears next to the caption in the menu item. On owner-draw menus, Bitmap can be used from an OnDrawItem (see page 1422) event handler to represent the entire menu item or a portion of the menu item.

Note: If the ImageIndex (see page 1411) property is set and the parent menu has a non-nil (Delphi) or NULL (C++) Images property, the image identified by ImageIndex (see page 1411) is used instead of Bitmap.

See Also

ImageIndex (see page 1411)

Images (see page 1345)

Caption (see page 1396)

OwnerDraw (see page 1350)

OnDrawItem (see page 1422)

34.1.5.5.7 TMenuItem.Break Property

Determines whether the menu item starts a new column in the menu.

Pascal

```
property Break: TMenuBreak;
```

C++

```
__property TMenuBreak Break;
```

Description

Set Break to break a long menu into columns at the item described by this menu item. The value of Break indicates whether the menu is broken into columns, and if so, whether a bar appears between the columns of the parent menu.

The default is mbNone (no breaking into columns).

See Also

Caption (see page 1396)

Parent (see page 1414)

34.1.5.5.8 TMenuItem.Caption Property

Specifies the text of the menu item.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Set Caption to the string the user sees for this item in the menu. To underline a character in a string, include an ampersand (&) before the character. This type of character is called an accelerator character. If Caption includes an accelerator character, the user can select the menu item by pressing Alt while typing the underlined character.

Tip: To automatically assign or adjust the accelerator characters in menu item captions, use the RethinkHotkeys (see page 1388) method.

Use the Caption property to identify the menu item as a line that separates the menu into parts. Specify a hyphen character (-) as the value of Caption for the menu item to indicate that the menu item is a separator. To separate the menu into multiple columns, use the Break (see page 1396) property instead.

Note: This property is not stored with the menu item if it is supplied by a TAction object.

C++ Examples:

```
/*
This example assumes that the main form of the application
has a main menu with a menu item. The following code adds a
separator, and the name of all forms to the menu. Do not
access FormCount in the mainform's FormCreate.  The other
forms do not exist yet!
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMenuItem *NewItem;
  // first create the separator
  NewItem = new TMenuItem(this); // If you change the AOwner (this) to MainMenu1->Items[1] and
then you add the new menu you will get "Menu inserted twice"?  Doesn't happen in Delphi.
  NewItem->Name = "Separator";   // The owner will cleanup these menu items.
  NewItem->Caption = "-";
  // add the new item to the Windows menu
  MainMenu1->Items->Items[1]->Add(NewItem);
  // now create and add a menu item for each form
  for  (int I = 0; I < Screen->FormCount; I++)
  {
    NewItem = new TMenuItem(this); // The owner will cleanup these menu items.
    NewItem->Caption = Screen->Forms[I]->Name + "Item";
    NewItem->Name = Screen->Forms[I]->Name;
    MainMenu1->Items->Items[1]->Add(NewItem);
  }
}
/*
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
```

```
    {
        switch (ClickedItem->Tag)
        {
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
            }
        }
    }
}

/*
To edit or destroy an item, grab its pointer via the Items
property.
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}

void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
This example assumes that the main form of the application
has a main menu with a menu item. The following code adds a
separator, and the name of all forms to the menu. Do not
access FormCount in the mainform's FormCreate.  The other
```

```
forms do not exist yet!
}
procedure TForm1.Button1Click(Sender: TObject);
var
  NewItem: TMenuItem;
  I : integer;
begin
  { first create the separator }
  NewItem := TMenuItem.Create(MainMenu1.Items[1]);
  NewItem.Name:= 'Separator';
  NewItem.Caption := '-';
  { add the new item to the Windows menu }
  MainMenu1.Items[1].Add(NewItem);
  { now create and add a menu item for each form }
  for  I := 0 to Screen.FormCount-1 do
  begin
    NewItem := TMenuItem.Create(MainMenu1.Items[1]);
    NewItem.Caption := Screen.Forms[I].Name + 'Item';
    NewItem.Name := Screen.Forms[I].Name;
    MainMenu1.Items[1].Add(NewItem);
   end;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  I : Integer;
begin
  // Count down, not up!
  for  I := MainMenu1.Items[1].Count - 1 downto 0 do
  begin
    MainMenu1.Items[1].Items[I].Free;
  end;
end;
{
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
}

type
  TForm1 = class(TForm)
    AddButton: TButton;
    EditButton: TButton;
    DestroyButton: TButton;
    PopupMenu1: TPopupMenu;
    procedure AddButtonClick(Sender: TObject);
    procedure EditButtonClick(Sender: TObject);
    procedure DestroyButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    procedure PopupMenuItemsClick(Sender: TObject);

  end;

var
  Form1: TForm1;

implementation
```

```
{$R *.dfm}

procedure TForm1.AddButtonClick(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
    // The owner (PopupMenu1) will cleanup this menu item.
  NewItem := TMenuItem.Create(PopupMenu1); // create the new item
  index := PopupMenu1.Items.Count;
  PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
  NewItem.Caption := 'Menu Item ' + IntToStr(index);
  NewItem.Tag := index;
  NewItem.OnClick :=
    PopupMenuItemsClick; // assign it an event handler
end;

procedure TForm1.PopupMenuItemsClick(Sender: TObject);
begin
  with Sender as TMenuItem do
  begin
    case Tag of
      0:  ShowMessage('first item clicked');
      1:  ShowMessage('second item clicked');
      2:  ShowMessage('third item clicked');
      3:  ShowMessage('fourth item clicked');
    end;
  end;
end;

{
To edit or destroy an item, grab its pointer via the Items
property.
}
procedure TForm1.EditButtonClick(Sender: TObject);
var
  ItemToEdit: TMenuItem;
begin
  ItemToEdit := PopupMenu.Items[1];
  ItemToEdit.Caption := 'Changed Caption';
end;

procedure TForm1.DestroyButtonClick(Sender: TObject);
var
  ItemToDelete: TMenuItem;
begin
  ItemToDelete := PopupMenu.Items[2];
  ItemToDelete.Free;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
  for index := 0 to 3 do
  begin
      // The owner (PopupMenu1) will cleanup this menu item.
    NewItem := TMenuItem.Create(PopupMenu1); // create the new item
    PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
    NewItem.Caption := 'Menu Item ' + IntToStr(index);
    NewItem.Tag := index;
    NewItem.OnClick :=
      PopupMenuItemsClick; // assign it an event handler
  end;
```

```
end;
```

See Also

Break (see page 1396)

Checked (see page 1401)

Caption

RethinkHotkeys (see page 1388)

34.1.5.5.9 TMenuItem.Checked Property

Specifies whether a check mark should appear beside the Caption (see page 1396).

Pascal

```
property Checked: Boolean;
```

C++

```
__property Boolean Checked;
```

Description

Use Checked to allow a menu item to work like a check box. If Checked is true, the menu item appears checked. If Checked is false, the menu item remains unchecked.

Note: This property is not stored with the menu item if it is supplied by a TAction object.

See Also

Caption (see page 1396)

RadioItem (see page 1414)

Checked

AutoCheck (see page 1394)

34.1.5.5.10 TMenuItem.Command Property

Specifies the Windows Command ID associated with the menu item.

Pascal

```
property Command: Word;
```

C++

```
__property Word Command;
```

Description

Use Command in an application that directly handles WM_COMMAND (see page 1684) messages. When the user selects the menu item, Windows sends a WM_COMMAND (see page 1684) to the window associated with the menu. The ItemID of the message contains the value of the Command property of the menu item that was selected.

See Also

OnClick (see page 1417)

DispatchCommand (see page 1335)

WindowHandle (see page 1351)

34.1.5.5.11 TMenuItem.Count Property

Indicates the number of subitems of the menu item.

Pascal

```
property Count: Integer;
```

C++

```
__property int Count;
```

Description

Read Count to determine the number of subitems listed in the Items (see page 1411) property array. When the user clicks on a menu item that has subitems, a dropdown menu appears which displays those subitems. Each subitem can, in turn, contain additional subitems. The Count property counts only the immediate subitems of the menu item.

C++ Examples:

```
/*
The following code disables all the subitems of of the menu
item File1.  This example requires a button, a TMainMenu
with several menu items added, one of them named File1.
*/
void __fastcall TForm1::DisableSubItemsClick(TObject *Sender)
{
  for (int i = 0; i < File1->Count; i++)
  {
    File1->Items[i]->Enabled = false;
  }
}

void __fastcall TForm1::Edit1Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the Edit command");
}

void __fastcall TForm1::File1Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the File command");
}

void __fastcall TForm1::FileSub11Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the FileSub1 command");
}

void __fastcall TForm1::FileSub21Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the FileSub2 command");
}

void __fastcall TForm1::FileSub31Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the FileSub3 command");
}

void __fastcall TForm1::Options1Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the Options command");
}
/*
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
```

```
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
    {
        switch (ClickedItem->Tag)
        {
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
            }
        }
    }
}

/*
To edit or destroy an item, grab its pointer via the Items
property.
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}

void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
```

```
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
The following code disables all the subitems of of the menu
item File1.  This example requires a button, a TMainMenu
with several menu items added, one of them named File1.
}
procedure TForm1.DisableSubItemsClick(Sender: TObject);
var
  I: Integer;
begin
  for I := 0 to File1.Count - 1 do
    File1.Items[I].Enabled := False;
end;

procedure TForm1.Edit1Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the Edit command');
end;

procedure TForm1.File1Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the File command');
end;

procedure TForm1.FileSub11Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the FileSub1 command');
end;

procedure TForm1.FileSub21Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the FileSub2 command');
end;

procedure TForm1.FileSub31Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the FileSub3 command');
end;

procedure TForm1.Options1Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the Options command');
end;
{
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
```

```
  Put the TPopupMenu in the PopupMenu property of the form.
  Place the PopupMenuItemsClick procedure in the TForm1 type
  declaration so that it can be used as the method call for
  the menu item OnClick event.
  }

  type
    TForm1 = class(TForm)
      AddButton: TButton;
      EditButton: TButton;
      DestroyButton: TButton;
      PopupMenu1: TPopupMenu;
      procedure AddButtonClick(Sender: TObject);
      procedure EditButtonClick(Sender: TObject);
      procedure DestroyButtonClick(Sender: TObject);
      procedure FormCreate(Sender: TObject);
    private
      { Private declarations }
    public
      procedure PopupMenuItemsClick(Sender: TObject);

    end;

  var
    Form1: TForm1;

  implementation

  {$R *.dfm}

  procedure TForm1.AddButtonClick(Sender: TObject);
  var
    index: Integer;
    NewItem: TMenuItem;
  begin
      // The owner (PopupMenu1) will cleanup this menu item.
    NewItem := TMenuItem.Create(PopupMenu1); // create the new item
    index := PopupMenu1.Items.Count;
    PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
    NewItem.Caption := 'Menu Item ' + IntToStr(index);
    NewItem.Tag := index;
    NewItem.OnClick :=
      PopupMenuItemsClick; // assign it an event handler
  end;

  procedure TForm1.PopupMenuItemsClick(Sender: TObject);
  begin
    with Sender as TMenuItem do
    begin
      case Tag of
        0:  ShowMessage('first item clicked');
        1:  ShowMessage('second item clicked');
        2:  ShowMessage('third item clicked');
        3:  ShowMessage('fourth item clicked');
      end;
    end;
  end;

  {
  To edit or destroy an item, grab its pointer via the Items
  property.
  }
  procedure TForm1.EditButtonClick(Sender: TObject);
  var
    ItemToEdit: TMenuItem;
  begin
```

```
    ItemToEdit := PopupMenu.Items[1];
    ItemToEdit.Caption := 'Changed Caption';
  end;

  procedure TForm1.DestroyButtonClick(Sender: TObject);
  var
    ItemToDelete: TMenuItem;
  begin
    ItemToDelete := PopupMenu.Items[2];
    ItemToDelete.Free;
  end;

  procedure TForm1.FormCreate(Sender: TObject);
  var
    index: Integer;
    NewItem: TMenuItem;
  begin
    for index := 0 to 3 do
    begin
        // The owner (PopupMenu1) will cleanup this menu item.
      NewItem := TMenuItem.Create(PopupMenu1); // create the new item
      PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
      NewItem.Caption := 'Menu Item ' + IntToStr(index);
      NewItem.Tag := index;
      NewItem.OnClick :=
        PopupMenuItemsClick; // assign it an event handler
    end;
  end;
```

See Also

Items (see page 1411)

MenuIndex (see page 1413)

34.1.5.5.12 TMenuItem.Default Property

Specifies whether the menu item is invoked when the parent item is double clicked.

Pascal

```
property Default: Boolean;
```

C++

```
__property Boolean Default;
```

Description

Set Default to specify whether a menu item is the default item in a submenu. Default menu items are executed when the parent item is double-clicked, allowing users to avoid navigating the submenus that contain them.

The default item appears in boldface. A submenu can have only one default item. If the Items (see page 1411) list of the parent menu item already contains a default menu item, setting the Default property to true sets the Default property of the previous default item to false.

See Also

Click (see page 1375)

Parent (see page 1414)

34.1.5.5.13 TMenuItem.Enabled Property

Specifies whether the menu item is enabled.

Pascal

```
property Enabled: Boolean;
```

C++

```
__property Boolean Enabled;
```

Description

Use Enabled to enable or disable a menu item. If Enabled is true, the Click (see page 1375) method is called when the user selects the item with the mouse. If Enabled is false, the menu item appears dimmed and the user cannot select it. However, calling the Click (see page 1375) method works even when Enabled is false.

Note: This property is not stored with the menu item if it is supplied by a TAction object.

C++ Examples:

```
/*
The following code disables all the subitems of of the menu
item File1.  This example requires a button, a TMainMenu
with several menu items added, one of them named File1.
*/
void __fastcall TForm1::DisableSubItemsClick(TObject *Sender)
{
  for (int i = 0; i < File1->Count; i++)
  {
    File1->Items[i]->Enabled = false;
  }
}

void __fastcall TForm1::Edit1Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the Edit command");
}

void __fastcall TForm1::File1Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the File command");
}

void __fastcall TForm1::FileSub11Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the FileSub1 command");
}

void __fastcall TForm1::FileSub21Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the FileSub2 command");
}

void __fastcall TForm1::FileSub31Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the FileSub3 command");
}

void __fastcall TForm1::Options1Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the Options command");
}
```

Delphi Examples:

```
{
The following code disables all the subitems of of the menu
```

```
item File1.  This example requires a button, a TMainMenu
with several menu items added, one of them named File1.
}
procedure TForm1.DisableSubItemsClick(Sender: TObject);
var
  I: Integer;
begin
  for I := 0 to File1.Count - 1 do
    File1.Items[I].Enabled := False;
end;

procedure TForm1.Edit1Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the Edit command');
end;

procedure TForm1.File1Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the File command');
end;

procedure TForm1.FileSub11Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the FileSub1 command');
end;

procedure TForm1.FileSub21Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the FileSub2 command');
end;

procedure TForm1.FileSub31Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the FileSub3 command');
end;

procedure TForm1.Options1Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the Options command');
end;
```

See Also

Click (see page 1375)

Visible (see page 1416)

Enabled

34.1.5.5.14 TMenuItem.GroupIndex Property

Identifies the logical group to which the menu item belongs.

Pascal

```
property GroupIndex: Byte;
```

C++

```
__property Byte GroupIndex;
```

Description

Use GroupIndex to control how menus are merged. In applications that contain multiple forms, items from the menus of secondary forms can be merged into the application's main menu as those forms become active. MDI applications always merge the menus of child windows with the main menu of the parent window.

By default, all menu items in a menu bar (main menu) have the same GroupIndex value. To use GroupIndex to merge menus, set the GroupIndex value of each successive menu item to a value equal to or greater than that of the previous menu item.

When a menu item in a menu bar that is merged into the application's main menu has the same GroupIndex value as a menu item in the main menu, the menu item in the application's main menu is replaced by the corresponding menu item. If multiple menu items in the application's main menu have the same GroupIndex value, they can be replaced only by separate menu items from the other form's menu that also have repeated GroupIndex values. Thus, the first main menu item with a given value is replaced by the first item in the second form with that value, the second item in the main form is replaced by the second item in the second form, and so on.

When a menu item in a menu bar that is merged into the application's main menu has a GroupIndex that falls between the GroupIndex values of two adjacent items on the application's main menu, that menu item is inserted between the two adjacent items. If the item's GroupIndex value is greater than all other GroupIndex values in the application's main menu, the item appears at the end of the main menu. If the GroupIndex value is less than all other GroupIndex values in the application's main menu, the menu item appears at the beginning of the main menu bar.

When an object created by an OLE 2.0 server application is activated, the server can try to merge its menus with the menus of the container application. The GroupIndex property is used to replace up to three main menu items with items from the server, in the manner described above. The server application uses the following predefined GroupIndex values to locate menu items to replace:

Group	Index	Description
Edit	1	Server menu items for editing the active OLE object
View	3	Server menu items for modifying the view of the OLE object.
Help	5	Server menu items for accessing the server's online Help.

Note: See the documentation for the OLE server for information about whether it attempts menu merge during in-place activation.

GroupIndex can also be used to define groups of menu items that act like radio buttons. To allow menu items to behave like radio buttons, set the RadioItem (see page 1414) property to all menu items in the group to true, and assign each menu item the same value of GroupIndex. Checking any menu item in the group will then uncheck the others.

See Also

MenuIndex (see page 1413)

RadioItem (see page 1414)

AutoMerge (see page 1328)

Merge (see page 1325)

PopulateOle2Menu (see page 1326)

34.1.5.5.15 TMenuItem.Handle Property

Indicates the Windows menu handle of the drop-down menu associated with the menu item.

Pascal

```
property Handle: HMENU;
```

C++

```
__property HMENU Handle;
```

Description

Use Handle when calling a Windows API function that requires a menu handle. Handle is only valid if the Count (see page 1402) property is greater than 0.

See Also

Count (see page 1402)

Items (see page 1411)

34.1.5.5.16 TMenuItem.HelpContext Property

Specifies the help context ID associated with the menu item.

Pascal

```
property HelpContext: THelpContext;
```

C++

```
__property THelpContext HelpContext;
```

Description

Use HelpContext to associate the menu item with a screen in the Help system of the application. Each screen in the Help system should have a unique context ID. When a menu item is selected using the keyboard, pressing F1 displays a Help screen. Which Help screen appears depends on the value of the HelpContext property.

Note: This property is not stored with the menu item if it is supplied by a TAction object.

See Also

HelpFile

HelpContext

HelpContext

34.1.5.5.17 TMenuItem.Hint Property

Specifies the text string that can appear when the user moves the mouse pointer over a menu item.

Pascal

```
property Hint: string;
```

C++

```
__property UnicodeString Hint;
```

Description

Set Hint to a string that provides more information about the meaning of the menu item than the Caption (see page 1396). The hint text appears in the **Status Bar** when the user pauses with the mouse over the menu item if Help Hints are enabled (that is, if the Form's and the Application's ShowHint properties are true). It is also available for the code in the application's OnHint event handler.

The value of Hint can specify both a short value for the Help Hint window and a longer string to be used by the OnHint event handler. To provide both a short and a long hint, set Hint to the short string, followed by a vertical bar (|), followed by the long string.

Note: This property is not stored with the menu item if it is supplied by a TAction object.

34.1.5.5.18 TMenuItem.ImageIndex Property

Indicates which image maintained by the parent menu appears next to the menu item.

Pascal

```
property ImageIndex: TImageIndex;
```

C++

```
__property TImageIndex ImageIndex;
```

Description

Set ImageIndex to designate an image that should appear next to the caption when the menu item is drawn. On top-level menu items, ImageIndex specifies a zero-offset index into the Images property of the parent menu (TMenu (see page 1331) or TPopupMenu (see page 1430)). On submenu items, ImageIndex is the zero-offset index into the parent item's SubMenuImages (see page 1415) property.

If the parent menu or item does not maintain a list of images, use the Bitmap (see page 1395) property to provide an image for a single menu item. However, any image specified by ImageIndex takes precedence over the Bitmap (see page 1395) property. Bitmap (see page 1395) is only used when ImageIndex has a negative value or the parent's Images or SubMenuImages (see page 1415) property is nil (Delphi) or NULL (C++).

Note: This property is not stored with the menu item if it is supplied by a TAction object.

See Also

Images (see page 1345)

Bitmap (see page 1395)

ImageIndex

SubMenuImages (see page 1415)

34.1.5.5.19 TMenuItem.Items Property

Lists the menu items in the submenu of the menu item.

Pascal

```
property Items [Index: Integer]: TMenuItem;
```

C++

```
__property TMenuItem Items[int Index];
```

Description

Use Items to access to a subitem by its position in the list of subitems. The value of Index is the position of the subitem within the Items array. For example, if an application has a File drop-down menu that contains the menu items New, Open, and Save, in that order, the expression

```
FileMenu.Items[2]
FileMenu->Items[2]
```

refers to the Save command.

Note: In Delphi, Items is the default property for TMenuItem (see page 1363). This means you can leave off the property name. Thus, instead of writing FileMenu.Items[2], you can write FileMenu[2].

Note: In C++, Items can be accessed using the [] operator, to achieve an affect similar to the default property in Delphi.

C++ Examples:

```
/*
The following code disables all the subitems of of the menu
item File1.  This example requires a button, a TMainMenu
with several menu items added, one of them named File1.
*/
void __fastcall TForm1::DisableSubItemsClick(TObject *Sender)
{
  for (int i = 0; i < File1->Count; i++)
  {
    File1->Items[i]->Enabled = false;
  }
}

void __fastcall TForm1::Edit1Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the Edit command");
}

void __fastcall TForm1::File1Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the File command");
}

void __fastcall TForm1::FileSub11Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the FileSub1 command");
}

void __fastcall TForm1::FileSub21Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the FileSub2 command");
}

void __fastcall TForm1::FileSub31Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the FileSub3 command");
}

void __fastcall TForm1::Options1Click(TObject *Sender)
{
  ListBox1->Items->Add("This is the Options command");
}
```

Delphi Examples:

```
{
The following code disables all the subitems of of the menu
item File1.  This example requires a button, a TMainMenu
with several menu items added, one of them named File1.
}
procedure TForm1.DisableSubItemsClick(Sender: TObject);
var
  I: Integer;
begin
  for I := 0 to File1.Count - 1 do
    File1.Items[I].Enabled := False;
end;

procedure TForm1.Edit1Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the Edit command');
end;
```

```
procedure TForm1.File1Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the File command');
end;

procedure TForm1.FileSub11Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the FileSub1 command');
end;

procedure TForm1.FileSub21Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the FileSub2 command');
end;

procedure TForm1.FileSub31Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the FileSub3 command');
end;

procedure TForm1.Options1Click(Sender: TObject);
begin
  ListBox1.Items.Add('This is the Options command');
end;
```

See Also

Count (see page 1402)

MenuIndex (see page 1413)

Parent (see page 1414)

34.1.5.5.20 TMenuItem.MenuIndex Property

Indicates the index of the menu item within its parent menu.

Pascal

```
property MenuIndex: Integer;
```

C++

```
__property int MenuIndex;
```

Description

Use MenuIndex to locate the menu item in the Items (see page 1411) property of its parent menu item. Changing this value moves the menu item within its menu.

Note: MenuIndex may not correspond to the position of the item in the menu the user sees. The menu can include items that are not visible.

See Also

Count (see page 1402)

GroupIndex (see page 1408)

Items (see page 1411)

Parent (see page 1414)

Visible (see page 1416)

34.1.5.5.21 TMenuItem.Parent Property

Identifies the parent menu item of this menu item.

Pascal

```
property Parent: TMenuItem;
```

C++

```
__property TMenuItem Parent;
```

Description

Read Parent to determine the menu item that represents the dropdown menu that contains this menu item. If the menu item is a top level menu item in a main menu or pop-up menu, Parent is the value of that menu's Items (see page 1411) property.

See Also

Items (see page 1411)

MenuIndex (see page 1413)

Items (see page 1346)

34.1.5.5.22 TMenuItem.RadioItem Property

Specifies whether the menu item is mutually exclusive with other menu items in its group.

Pascal

```
property RadioItem: Boolean;
```

C++

```
__property Boolean RadioItem;
```

Description

Use RadioItem to make the menu item behave like a radio button with a group of other menu items. The group of menu items is the set of all menu items in a pop-up or drop-down menu that have the same value of GroupIndex (see page 1408). When RadioItem is true, only one menu item in the group that contains the menu item can be checked at a time. The single selected menu item in the group is indicated by a round dot next to the Caption (see page 1396).

Note: RadioItem does not automatically select the item when the user clicks it at runtime: you must still explicitly set the Checked (see page 1401) property to true in response to the OnClick (see page 1417) event. RadioItem simply changes the appearance of the Checked (see page 1401) state (from a check mark to a round dot) and automatically unchecks other menu items in the same group when you set the Checked (see page 1401) property to true.

RadioItem does not work for the top-level items in a main menu.

Note: Under Windows NT 3.51, the round dot indicator only appears if the "new shell" has been installed.

See Also

Caption (see page 1396)

Checked (see page 1401)

GroupIndex (see page 1408)

TMainMenu (see page 1322)

OnClick (see page 1417)

34.1.5.5.23 TMenuItem.ShortCut Property

Specifies the key combination users can type to quickly access the menu item.

Pascal

```
property ShortCut: TShortCut;
```

C++

```
__property TShortCut ShortCut;
```

Description

Use ShortCut to allow the user to type a key combination instead of selecting the item using the menu. This key combination appears to the right of the menu item in the menu.

When setting menu shortcuts at design time, the Object Inspector provides a long list of key combinations to choose from. To create menu shortcuts at runtime, use the global ShortCut or TextToShortCut (see page 1457) function. For example, the following line creates a shortcut, Ctrl+C and assigns it to a menu item named CloseCommand.

```
CloseCommand.ShortCut := ShortCut(Word('C'), [ssCtrl]);
CloseCommand->ShortCut = ShortCut(Word('C'), TShiftState() << ssCtrl);
```

Note: This property is not stored with the menu item if it is supplied by a TAction object.

See Also

ShortCut (see page 1453)

ShortCutToKey (see page 1454)

ShortCutToText (see page 1456)

TextToShortCut (see page 1457)

IsShortCut (see page 1339)

ShortCut

34.1.5.5.24 TMenuItem.SubMenuImages Property

Lists the images that can appear beside individual items in a submenu of this menu item.

Pascal

```
property SubMenuImages: TCustomImageList;
```

C++

```
__property TCustomImageList SubMenuImages;
```

Description

Use SubMenuImages to supply a set of images that can appear beside items listed in the Items (see page 1411) property. Individual menu items set their ImageIndex (see page 1411) property to indicate which image in the list appears to the left of the menu item caption.

Tip: If the application uses action lists, this property can be the value of the command list's Images property. This way, menu items and buttons that do the same thing can display the same image.

See Also

ImageIndex (see page 1411)

Images

Items (see page 1411)

34.1.5.5.25 TMenuItem.Visible Property

Specifies whether the menu item appears in the menu.

Pascal

```
property Visible: Boolean;
```

C++

```
__property Boolean Visible;
```

Description

Use Visible to prevent the user from seeing an item in the menu. For example, remove menu items that do not apply to the current component of a pop-up menu in an OnPopup event handler.

If Visible is true, the menu item appears. If Visible is false, the menu item is hidden.

Note: This property is not stored with the menu item if it is supplied by a TAction object.

See Also

Enabled (see page 1406)

OnPopup (see page 1444)

PopupComponent (see page 1440)

Visible

34.1.5.6 TMenuItem Events

The events of the TMenuItem class are listed here.

Published Events

	Name	Description
	OnAdvancedDrawItem (see page 1416)	Occurs when an owner-draw menu needs to be drawn.
	OnClick (see page 1417)	Occurs when the user clicks menu item.
	OnDrawItem (see page 1422)	Occurs when an owner-draw menu needs to be drawn.
	OnMeasureItem (see page 1423)	Occurs on owner-draw menu items to determine menu item size before the menu is drawn.

34.1.5.6.1 TMenuItem.OnAdvancedDrawItem Event

Occurs when an owner-draw menu needs to be drawn.

Pascal

```
property OnAdvancedDrawItem: TAdvancedMenuDrawItemEvent;
```

C++

```
__property TAdvancedMenuDrawItemEvent OnAdvancedDrawItem;
```

Description

Write an OnAdvancedDrawItem event handler to render the image of a menu item in an owner-draw menu.

In the event handler, use the Bitmap (see page 1395) property or the ImageIndex (see page 1411) property to access a bitmap that represents the menu item.

Note: OnAdvancedDrawItem will only occur if the parent menu's OwnerDraw property is true or the parent menu's Images property has been set.

See Also

OnMeasureItem (see page 1423)

Bitmap (see page 1395)

ImageIndex (see page 1411)

Images (see page 1345)

OwnerDraw (see page 1350)

OnDrawItem (see page 1422)

34.1.5.6.2 TMenuItem.OnClick Event

Occurs when the user clicks menu item.

Pascal

```
property OnClick: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnClick;
```

Description

Write an OnClick event handler to implement the desired behavior for when the user selects the menu item. An advantage of using the OnClick event handler over directly handling WM_COMMAND (see page 1684) messages sent when the user selects the menu item, is that only the OnClick event occurs when the Click (see page 1375) method is called.

Note: This event handler is not stored with the menu item if it is supplied by a TAction object.

C++ Examples:

```
/*
This example uses several forms. The first form has its
FormStyle property set to MDIForm. The others have their
FormStyle properties set to MDIChild and their Visible
properties set to true. Add a main menu component and name
one of the menu items MyArrangeIcons. This is code for the
MyArrangeIconsClick handler.  When the user chooses the
ArrangeIcons command, minimized child forms are arranged
so that they are evenly spaced and don't overlap.  DO NOT
name the menu items "ArrangeIcons" as that will override
the TForm method.
*/
void __fastcall TForm1::MyArrangeIconsClick(TObject *Sender)
{
  Form1->ArrangeIcons();
}
/*
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  PopupMenu1->AutoPopup = true;
  Edit1->PopupMenu = PopupMenu1;
```

```
  Edit2->PopupMenu = PopupMenu1;
}

/*
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
*/
void __fastcall TForm1::Copy1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CopyToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Cut1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CutToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Paste1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->PasteFromClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}
/*
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
```

```
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
    {
        switch (ClickedItem->Tag)
        {
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
            }
        }
    }
}

/*
To edit or destroy an item, grab its pointer via the Items
property.
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}

void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
```

```
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
This example uses several forms. The first form has its
FormStyle property set to MDIForm. The others have their
FormStyle properties set to MDIChild and their Visible
properties set to true. Add a main menu component and name
one of the menu items MyArrangeIcons. This is code for the
MyArrangeIconsClick handler.  When the user chooses the
ArrangeIcons command, minimized child forms are arranged
so that they are evenly spaced and don't overlap.  DO NOT
name the menu items "ArrangeIcons" as that will override
the TForm method.
}
procedure TForm1.MyArrangeIconsClick(Sender: TObject);
begin
  Form1.ArrangeIcons;
end;
{
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  PopupMenu1.AutoPopup := True;
  Edit1.PopupMenu := PopupMenu1;
  Edit2.PopupMenu := PopupMenu1;
end;

{
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
}
procedure TForm1.Copy1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
   Edit1.CopyToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CopyToClipboard
  else
    Beep;
end;

procedure TForm1.Cut1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.CutToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CutToClipboard
  else
    Beep;
```

```
end;

procedure TForm1.Paste1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.PasteFromClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.PasteFromClipboard
  else
    Beep;
end;
{
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
}

type
  TForm1 = class(TForm)
    AddButton: TButton;
    EditButton: TButton;
    DestroyButton: TButton;
    PopupMenu1: TPopupMenu;
    procedure AddButtonClick(Sender: TObject);
    procedure EditButtonClick(Sender: TObject);
    procedure DestroyButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    procedure PopupMenuItemsClick(Sender: TObject);

  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.AddButtonClick(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
    // The owner (PopupMenu1) will cleanup this menu item.
  NewItem := TMenuItem.Create(PopupMenu1); // create the new item
  index := PopupMenu1.Items.Count;
  PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
  NewItem.Caption := 'Menu Item ' + IntToStr(index);
  NewItem.Tag := index;
  NewItem.OnClick :=
    PopupMenuItemsClick; // assign it an event handler
end;

procedure TForm1.PopupMenuItemsClick(Sender: TObject);
begin
  with Sender as TMenuItem do
  begin
```

```
      case Tag of
        0:  ShowMessage('first item clicked');
        1:  ShowMessage('second item clicked');
        2:  ShowMessage('third item clicked');
        3:  ShowMessage('fourth item clicked');
      end;
    end;
  end;

  {
  To edit or destroy an item, grab its pointer via the Items
  property.
  }
  procedure TForm1.EditButtonClick(Sender: TObject);
  var
    ItemToEdit: TMenuItem;
  begin
    ItemToEdit := PopupMenu.Items[1];
    ItemToEdit.Caption := 'Changed Caption';
  end;

  procedure TForm1.DestroyButtonClick(Sender: TObject);
  var
    ItemToDelete: TMenuItem;
  begin
    ItemToDelete := PopupMenu.Items[2];
    ItemToDelete.Free;
  end;

  procedure TForm1.FormCreate(Sender: TObject);
  var
    index: Integer;
    NewItem: TMenuItem;
  begin
    for index := 0 to 3 do
    begin
        // The owner (PopupMenu1) will cleanup this menu item.
      NewItem := TMenuItem.Create(PopupMenu1); // create the new item
      PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
      NewItem.Caption := 'Menu Item ' + IntToStr(index);
      NewItem.Tag := index;
      NewItem.OnClick :=
        PopupMenuItemsClick; // assign it an event handler
    end;
  end;
```

See Also

Click (see page 1375)

Command (see page 1401)

OnExecute

34.1.5.6.3 TMenuItem.OnDrawItem Event

Occurs when an owner-draw menu needs to be drawn.

Pascal

```
property OnDrawItem: TMenuDrawItemEvent;
```

C++

```
__property TMenuDrawItemEvent OnDrawItem;
```

Description

Write an OnDrawItem event handler to render the image of a menu item in an owner-draw menu. In the event handler, use the Bitmap (see page 1395) property or the ImageIndex (see page 1411) property to access a bitmap that represents the menu item.

Note: OnDrawItem will only occur if the parent menu's OwnerDraw property is true or the parent menu's Images property has been set.

Tip: For more precise information about the menu item's state, use the OnAdvancedDrawItem (see page 1416) event instead.

See Also

OnMeasureItem (see page 1423)

Bitmap (see page 1395)

ImageIndex (see page 1411)

Images (see page 1345)

OwnerDraw (see page 1350)

OnAdvancedDrawItem (see page 1416)

34.1.5.6.4 TMenuItem.OnMeasureItem Event

Occurs on owner-draw menu items to determine menu item size before the menu is drawn.

Pascal

```
property OnMeasureItem: TMenuMeasureItemEvent;
```

C++

```
__property TMenuMeasureItemEvent OnMeasureItem;
```

Description

Write an OnMeasureItem event handler to indicate the size of owner-draw menu items. If the menu item is using a bitmap image, the event handler can use the Bitmap (see page 1395) property or the ImageIndex (see page 1411) property to access the bitmap to determine its size.

Note: OnMeasureItem will only occur if the parent menu's OwnerDraw property is true or the parent menu's Images property has been set.

See Also

OnDrawItem (see page 1422)

Bitmap (see page 1395)

ImageIndex (see page 1411)

Images (see page 1345)

OwnerDraw (see page 1350)

TextExtent

34.1.6 TMenuItemEnumerator Class

Class Hierarchy

Pascal

```
TMenuItemEnumerator = class;
```

C++

```
class TMenuItemEnumerator;
```

File

Menus

Description

This is class Menus.TMenuItemEnumerator.

34.1.6.1 TMenuItemEnumerator Members

The following tables list the members exposed by TMenuItemEnumerator.

Public Methods

	Name	Description
	Create (see page 1424)	This is Create, a member of class TMenuItemEnumerator.
	GetCurrent (see page 1425)	This is GetCurrent, a member of class TMenuItemEnumerator.
	MoveNext (see page 1425)	This is MoveNext, a member of class TMenuItemEnumerator.

Public Properties

	Name	Description
R	Current (see page 1425)	This is Current, a member of class TMenuItemEnumerator.

34.1.6.2 TMenuItemEnumerator.Create Constructor

Pascal

```
constructor Create(AMenuItem: TMenuItem);
```

C++

```
__fastcall TMenuItemEnumerator(TMenuItem AMenuItem);
```

Description

This is Create, a member of class TMenuItemEnumerator.

34.1.6.3 TMenuItemEnumerator Methods

The methods of the TMenuItemEnumerator class are listed here.

Public Methods

	Name	Description
	GetCurrent (see page 1425)	This is GetCurrent, a member of class TMenuItemEnumerator.
	MoveNext (see page 1425)	This is MoveNext, a member of class TMenuItemEnumerator.

34.1.6.3.1 TMenuItemEnumerator.GetCurrent Method

Pascal

```
function GetCurrent: TMenuItem;
```

C++

```
__fastcall TMenuItem GetCurrent();
```

Description

This is GetCurrent, a member of class TMenuItemEnumerator.

34.1.6.3.2 TMenuItemEnumerator.MoveNext Method

Pascal

```
function MoveNext: Boolean;
```

C++

```
__fastcall Boolean MoveNext();
```

Description

This is MoveNext, a member of class TMenuItemEnumerator.

34.1.6.4 TMenuItemEnumerator Properties

The properties of the TMenuItemEnumerator class are listed here.

Public Properties

	Name	Description
	Current (see page 1425)	This is Current, a member of class TMenuItemEnumerator.

34.1.6.4.1 TMenuItemEnumerator.Current Property

Pascal

```
property Current: TMenuItem;
```

C++

```
__property TMenuItem Current;
```

Description

This is Current, a member of class TMenuItemEnumerator.

34.1.7 TMenuItemStack Class

Class Hierarchy

Pascal

```
TMenuItemStack = class(TStack);
```

C++

```
class TMenuItemStack : public TStack;
```

File

Menus

Description

This is class Menus.TMenuItemStack.

34.1.7.1 TMenuItemStack Members

The following tables list the members exposed by TMenuItemStack.

Public Methods

	Name	Description
	ClearItem (see page 1426)	This is ClearItem, a member of class TMenuItemStack.

34.1.7.2 TMenuItemStack Methods

The methods of the TMenuItemStack class are listed here.

Public Methods

	Name	Description
	ClearItem (see page 1426)	This is ClearItem, a member of class TMenuItemStack.

34.1.7.2.1 TMenuItemStack.ClearItem Method

Pascal

```
procedure ClearItem(AItem: TMenuItem);
```

C++

```
__fastcall ClearItem(TMenuItem AItem);
```

Description

This is ClearItem, a member of class TMenuItemStack.

34.1.8 TPopupList Class

TPopupList manages the window that handles popup menu messages.

Class Hierarchy

Pascal

```
TPopupList = class(TList);
```

C++

```
class TPopupList : public TList;
```

File

Menus

Description

TPopupList is the type of the global PopupList (see page 1470) variable, which provides centralized message handling for all popup menus in an application.

See Also

TPopupMenu (see page 1430)

PopupList (see page 1470)

34.1.8.1 TPopupList Members

The following tables list the members exposed by TPopupList.

Protected Fields

	Name	Description
	FWindow (see page 1428)	This is FWindow, a member of class TPopupList.

Protected Methods

	Name	Description
	MainWndProc (see page 1429)	Wraps the WndProc (see page 1429) method with an exception-handling layer.
V	WndProc (see page 1429)	Provides centralized message handling for all popup menus in the application.

Public Methods

	Name	Description
	Add (see page 1428)	Inserts a new popup menu at the end of the list.
	Remove (see page 1429)	Removes a popup menu from the list.

Public Properties

	Name	Description
R	Window (see page 1430)	Provides access to the Window handle of the hidden window that processes popup menu messages.

34.1.8.2 TPopupList Fields

The fields of the TPopupList class are listed here.

Protected Fields

	Name	Description
	FWindow (see page 1428)	This is FWindow, a member of class TPopupList.

34.1.8.2.1 TPopupList.FWindow Field

Pascal

```
FWindow: HWND;
```

C++

```
HWND FWindow;
```

Description

This is FWindow, a member of class TPopupList.

34.1.8.3 TPopupList Methods

The methods of the TPopupList class are listed here.

Protected Methods

	Name	Description
	MainWndProc (see page 1429)	Wraps the WndProc (see page 1429) method with an exception-handling layer.
V	WndProc (see page 1429)	Provides centralized message handling for all popup menus in the application.

Public Methods

	Name	Description
	Add (see page 1428)	Inserts a new popup menu at the end of the list.
	Remove (see page 1429)	Removes a popup menu from the list.

34.1.8.3.1 TPopupList.Add Method

Inserts a new popup menu at the end of the list.

Pascal

```
procedure Add(Popup: TPopupMenu);
```

C++

```
__fastcall Add(TPopupMenu Popup);
```

Description

Do not call the Add method. Popup menus automatically add themselves to the list when they are created.

Add increments Count and, if necessary, allocates memory by increasing the value of Capacity. When the first popup menu is added to the list, Add creates the hidden window whose handle is available as the Window (see page 1430) property.

See Also

Remove (see page 1429)

Window (see page 1430)

34.1.8.3.2 TPopupList.MainWndProc Method

Wraps the WndProc (see page 1429) method with an exception-handling layer.

Pascal

```
procedure MainWndProc(var Message: TMessage);
```

C++

```
__fastcall MainWndProc(TMessage Message);
```

Description

Do not use the protected MainWndProc method. It is used internally when creating the hidden window that responds to popup menu messages. To subclass the window procedure that handles messages sent to popup menus, override the WndProc (see page 1429) method instead.

See Also

WndProc (see page 1429)

34.1.8.3.3 TPopupList.Remove Method

Removes a popup menu from the list.

Pascal

```
procedure Remove(Popup: TPopupMenu);
```

C++

```
__fastcall Remove(TPopupMenu Popup);
```

Description

Do not call the Remove method. It is called automatically by popup menus from their destructors, to remove themselves from the list.

See Also

Add (see page 1428)

34.1.8.3.4 TPopupList.WndProc Method

Provides centralized message handling for all popup menus in the application.

Pascal

```
procedure WndProc(var Message: TMessage); virtual;
```

C++

```
virtual __fastcall WndProc(TMessage Message);
```

Description

Override WndProc to subclass the window procedure that handles messages directed at the popup menus in an application. WndProc handles messages by dispatching commands, displaying help, painting menu item, and so on.

34.1.8.4 TPopupList Properties

The properties of the TPopupList class are listed here.

Public Properties

	Name	Description
	Window (see page 1430)	Provides access to the Window handle of the hidden window that processes popup menu messages.

34.1.8.4.1 TPopupList.Window Property

Provides access to the Window handle of the hidden window that processes popup menu messages.

Pascal

```
property Window: HWND;
```

C++

```
__property HWND Window;
```

Description

Use Window for Windows API calls that require a window handle. The hidden window that processes messages for the popup menus in an application is not created until the first popup menu is added to the list. If the list contains no popup menus (that is, if no popup menus are created by the application), Window is 0.

34.1.9 TPopupMenu Class

TPopupMenu encapsulates the properties, methods, and events of a pop-up menu.

Class Hierarchy

Pascal

```
TPopupMenu = class(TMenu);
```

C++

```
class TPopupMenu : public TMenu;
```

File

Menus

Description

Use TPopupMenu to define the pop-up menu that appears when the user clicks on a control with the right mouse button. To make a pop-up menu available, assign the TPopupMenu object to the control's PopupMenu property.

Tip: To create a popup menu that users can configure at runtime, use the TStandardMenuPopup component with an action manager instead.

Note: If the popup menu's ParentBiDiMode (see page 1440) is true, the popup menu's BiDiMode (see page 1438) is set to the BiDiMode (see page 1438) of the control that activates it. If a control cannot be found, the BiDiMode (see page 1438) is

set to the value of the Application object's BiDiMode (see page 1438). The popup menu's BiDiMode (see page 1438) affects all of its menu items.

See Also

TMainMenu (see page 1322)

TMenuItem (see page 1363)

NewPopupMenu (see page 1451)

TStandardMenuPopup

TActionManager

Providing a Pop-up Menu

Setting Up Action Bands

34.1.9.1 TPopupMenu Members

The following tables list the members exposed by TPopupMenu.

Protected Methods

	Name	Description
	AdjustBiDiBehavior (see page 1335)	Applies the BiDiMode (see page 1344) property to the menu when the Windows control is created.
	DoBiDiModeChanged (see page 1336)	Updates the menu when the BiDiMode (see page 1344) property changes.
	DoChange (see page 1336)	Generates an OnChange (see page 1352) event.
	DoGetMenuString (see page 1337)	Returns the caption for an item in the menu.
	DoPopup (see page 1433)	Generates an OnPopup (see page 1444) event.
	GetHandle (see page 1338)	Returns the value of the Handle (see page 1345) property.
	IsOwnerDraw (see page 1339)	Indicates whether menu items can contain more than simple captions.
	Loaded (see page 1340)	Calls the DoChange (see page 1336) method when the menu is first loaded into memory.
	MenuChanged (see page 1340)	Responds when the menu changes.
	Notification (see page 1341)	Responds when components are added or deleted.
	SetChildOrder (see page 1342)	Ensures that the menu item indexes reflect the order in which they are loaded and saved.
	SetPopupPoint (see page 1434)	This is SetPopupPoint, a member of class TPopupMenu.
	UpdateItems (see page 1343)	Causes all menu items to update themselves.
	UseRightToLeftAlignment (see page 1435)	Specifies whether the control's alignment is in a right-to-left mode.

Public Methods

	Name	Description
	Create (see page 1432)	Creates an instance of TPopupMenu (see page 1430).
	CreateMenuItem (see page 1335)	This is CreateMenuItem, a member of class TMenu.
	Destroy (see page 1433)	Destroys an instance of TPopupMenu (see page 1430).
	DispatchCommand (see page 1335)	Calls the Click method of the menu item with the specified value as its Command property.
	DispatchPopup (see page 1336)	Dispatches a pop-up menu command.
	FindItem (see page 1337)	Locates and returns a menu item.
	GetChildren (see page 1338)	Executes a callback for child menu items.
	GetHelpContext (see page 1338)	Returns the help context ID associated with a menu item.
	IsRightToLeft (see page 1339)	Indicates whether menu items read from right to left.
	IsShortCut (see page 1339)	Dispatches a key message if it represents a shortcut key for an item in the menu.
	ParentBiDiModeChanged (see page 1341)	This is the overview for the ParentBiDiModeChanged method overload.
	Popup (see page 1434)	Displays the pop-up menu onscreen.

	ProcessMenuChar (see page 1342)	Responds to keyboard input for the menu.

Public Properties

	Name	Description
R	Handle (see page 1345)	Provides access to the Windows menu handle for the menu.
	PopupComponent (see page 1440)	Indicates the component that last displayed the popup menu in response to a right mouse click.
R	PopupPoint (see page 1443)	Indicates the location where the pop-up menu appears.
	WindowHandle (see page 1351)	Provides access to the handle of the window that uses the menu.

Published Events

	Name	Description
	OnChange (see page 1443)	Occurs when the menu changes.
	OnPopup (see page 1444)	Occurs just before the pop-up menu appears.

Published Properties

	Name	Description
	Alignment (see page 1436)	Determines where the pop-up menu appears when the user clicks the right mouse button.
	AutoHotkeys (see page 1436)	Determines whether the accelerator keys for menu items can be reset automatically.
	AutoLineReduction (see page 1437)	Determines whether redundant separator bars are automatically removed from the menu.
	AutoPopup (see page 1437)	Determines whether the pop-up menu appears automatically when the user clicks the right mouse button.
	BiDiMode (see page 1438)	Determines the layout of the menu in Middle East locales.
	HelpContext (see page 1438)	Specifies a context ID for the entire pop-up menu.
	Images (see page 1439)	Lists the images that can appear beside individual menu items.
R	Items (see page 1346)	Describes the elements of the menu.
	MenuAnimation (see page 1439)	Specifies how the menu appears when it is first displayed.
	OwnerDraw (see page 1440)	Specifies whether the menu items are drawn by the application.
	ParentBiDiMode (see page 1440)	Indicates whether the BiDiMode property is inherited from the control that uses the menu.
	TrackButton (see page 1443)	Specifies which mouse button activates the popup menu when it is associated with a toolbar button.

34.1.9.2 TPopupMenu.Create Constructor

Creates an instance of TPopupMenu (see page 1430).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TPopupMenu(TComponent * AOwner);
```

Description

Call Create to create to create a pop-up menu at runtime. Pop-up menus placed in forms or data modules at design time are created automatically.

See Also

Create (see page 1333)

34.1.9.3 TPopupMenu.Destroy Destructor

Destroys an instance of TPopupMenu (see page 1430).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TPopupMenu();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TPopupMenu (see page 1430) reference is not nil, and only then calls Destroy.

See Also

System::TObject::Free

34.1.9.4 TPopupMenu Methods

The methods of the TPopupMenu class are listed here.

Protected Methods

	Name	Description
	DoPopup (see page 1433)	Generates an OnPopup (see page 1444) event.
	SetPopupPoint (see page 1434)	This is SetPopupPoint, a member of class TPopupMenu.
	UseRightToLeftAlignment (see page 1435)	Specifies whether the control's alignment is in a right-to-left mode.

Public Methods

	Name	Description
	Popup (see page 1434)	Displays the pop-up menu onscreen.

34.1.9.4.1 TPopupMenu.DoPopup Method

Generates an OnPopup (see page 1444) event.

Pascal

```
procedure DoPopup(Sender: TObject); virtual;
```

C++

```
virtual __fastcall DoPopup(TObject * Sender);
```

Description

DoPopup is called automatically just before the pop-up menu appears to generate an OnPopup (see page 1444) event. Override DoPopup in a descendant class to change the event.

See Also

OnPopup (see page 1444)

Popup (see page 1434)

34.1.9.4.2 TPopupMenu.Popup Method

Displays the pop-up menu onscreen.

Pascal

```
procedure Popup(X: Integer; Y: Integer); virtual;
```

C++

```
virtual __fastcall Popup(int X, int Y);
```

Description

Call Popup to bring up the pop-up menu. The menu appears at the screen coordinates indicated by the values (in pixels) of X and Y.

C++ Examples:

```
/*
This example uses a pop-up menu. When the user presses the
mouse button, the pop-up menu appears where the user
clicked the mouse:
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  PopupMenu1->AutoPopup = false;
}

void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
          int X, int Y)
{
  PopupMenu1->Popup(Left + X, Top + Y);
}
```

Delphi Examples:

```
{
This example uses a pop-up menu. When the user presses the
mouse button, the pop-up menu appears where the user
clicked the mouse:
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  PopupMenu1.AutoPopup := False;
end;

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  PopupMenu1.Popup(Left + X, Top + Y);
end;
```

See Also

Alignment (see page 1436)

AutoPopup (see page 1437)

OnPopup (see page 1444)

34.1.9.4.3 TPopupMenu.SetPopupPoint Method

Pascal

```
procedure SetPopupPoint(APopupPoint: TPoint);
```

C++

```
__fastcall SetPopupPoint(TPoint APopupPoint);
```

Description

This is SetPopupPoint, a member of class TPopupMenu.

34.1.9.4.4 TPopupMenu.UseRightToLeftAlignment Method

Specifies whether the control's alignment is in a right-to-left mode.

Pascal

```
function UseRightToLeftAlignment: Boolean;
```

C++

```
__fastcall Boolean UseRightToLeftAlignment();
```

Description

Call UseRightToLeftAlignment to determine whether the control's alignment is in a right-to-left mode. This information can be used when painting the control or when responding to mouse events.

UseRightToLeftAlignment returns the setting from the control that activated the popup. If the popup was not activated by a control, UseRightToLeftAlignment returns the setting used in the global Application object.

34.1.9.5 TPopupMenu Properties

The properties of the TPopupMenu class are listed here.

Public Properties

	Name	Description
	PopupComponent (see page 1440)	Indicates the component that last displayed the popup menu in response to a right mouse click.
R	PopupPoint (see page 1443)	Indicates the location where the pop-up menu appears.

Published Properties

	Name	Description
	Alignment (see page 1436)	Determines where the pop-up menu appears when the user clicks the right mouse button.
	AutoHotkeys (see page 1436)	Determines whether the accelerator keys for menu items can be reset automatically.
	AutoLineReduction (see page 1437)	Determines whether redundant separator bars are automatically removed from the menu.
	AutoPopup (see page 1437)	Determines whether the pop-up menu appears automatically when the user clicks the right mouse button.
	BiDiMode (see page 1438)	Determines the layout of the menu in Middle East locales.
	HelpContext (see page 1438)	Specifies a context ID for the entire pop-up menu.
	Images (see page 1439)	Lists the images that can appear beside individual menu items.
	MenuAnimation (see page 1439)	Specifies how the menu appears when it is first displayed.
	OwnerDraw (see page 1440)	Specifies whether the menu items are drawn by the application.
	ParentBiDiMode (see page 1440)	Indicates whether the BiDiMode property is inherited from the control that uses the menu.
	TrackButton (see page 1443)	Specifies which mouse button activates the popup menu when it is associated with a toolbar button.

34.1.9.5.1 TPopupMenu.Alignment Property

Determines where the pop-up menu appears when the user clicks the right mouse button.

Pascal

```
property Alignment: TPopupAlignment;
```

C++

```
__property TPopupAlignment Alignment;
```

Description

Set Alignment to control the placement of the popup menu. Alignment controls whether the menu appears to the left of the mouse, to the right of the mouse, or centered over the mouse.

By default, the menu appears with its top left corner under the mouse (paLeft).

See Also

AutoPopup (see page 1437)

OnPopup (see page 1444)

34.1.9.5.2 TPopupMenu.AutoHotkeys Property

Determines whether the accelerator keys for menu items can be reset automatically.

Pascal

```
property AutoHotkeys: TMenuAutoFlag;
```

C++

```
__property TMenuAutoFlag AutoHotkeys;
```

Description

Set AutoHotkeys to indicate whether the menu should automatically ensure that accelerator keys for its items are reset if necessary so that every menu item has an accelerator key and no two items have the same accelerator key. This ensures that menu items added dynamically at runtime can have accelerator keys that do not introduce conflicts. TMenu tries to preserve all existing accelerator key mappings when adjusting the accelerator keys, but may be forced to change some in order to ensure that every item has a unique accelerator.

Before the menu is displayed, the menu checks and adjusts the accelerator key mappings if AutoHotkeys is maAutomatic. When AutoHotkeys is maManual, the menu does not alter the top-level menu items, but may adjust the accelerator keys for submenus, depending on the AutoHotkeys property of the menu item that is the parent of the submenu.

When AutoHotkeys is maManual, an application can still adjust the menu items by calling the Items property's RethinkHotkeys method. AutoHotkeys merely controls whether this adjustment occurs automatically.

Note: The AutoHotkeys property of a menu is linked to the AutoHotkeys property of the TMenuItems object that implements its Items property. Setting one sets the other.

See Also

AutoHotkeys (see page 1394)

Items (see page 1346)

RethinkHotkeys (see page 1388)

34.1.9.5.3 TPopupMenu.AutoLineReduction Property

Determines whether redundant separator bars are automatically removed from the menu.

Pascal

```
property AutoLineReduction: TMenuAutoFlag;
```

C++

```
__property TMenuAutoFlag AutoLineReduction;
```

Description

Set AutoLineReduction to indicate whether the menu should automatically ensure that the menu does not start or end with a separator bar and that two separator bars do not appear next to each other. This ensures that menus built dynamically at runtime do not have separator bars in undesired positions.

Before the menu is displayed, it removes superfluous separator bars when AutoLineReduction is maAutomatic. When AutoLineReduction is maManual, the menu does not remove separators from the top-level menu items, but may remove superfluous separators from submenus, depending on the AutoLineReduction property of the menu item that is the parent of the submenu.

When AutoLineReduction is maManual, an application can still remove superfluous separators by calling the Items property's RethinkLines method. AutoLineReduction merely determines whether this adjustment occurs automatically.

Note: The AutoLineReduction property of a menu is linked to the AutoLineReduction property of the TMenuItems object that implements its Items property. Setting one sets the other.

See Also

AutoLineReduction (see page 1395)

Items (see page 1346)

RethinkLines (see page 1389)

34.1.9.5.4 TPopupMenu.AutoPopup Property

Determines whether the pop-up menu appears automatically when the user clicks the right mouse button.

Pascal

```
property AutoPopup: Boolean;
```

C++

```
__property Boolean AutoPopup;
```

Description

Set AutoPopup to true to cause the pop-up menu to appear automatically when the user clicks the right mouse button on a control that has the pop-up menu specified as its PopupMenu property. The application key present on keyboards specifically designed for 32-bit Windows will also show this menu when pressed.

Set AutoPopup to false if the appearance of the pop-up menu should be controlled programmatically. To display a pop-up menu when AutoPopup is false, use the Popup (see page 1434) method.

C++ Examples:

```
/*
This example uses a pop-up menu. When the user presses the
mouse button, the pop-up menu appears where the user
clicked the mouse:
```

```
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  PopupMenu1->AutoPopup = false;
}

void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
          int X, int Y)
{
  PopupMenu1->Popup(Left + X, Top + Y);
}
```

Delphi Examples:

```
{
This example uses a pop-up menu. When the user presses the
mouse button, the pop-up menu appears where the user
clicked the mouse:
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  PopupMenu1.AutoPopup := False;
end;

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  PopupMenu1.Popup(Left + X, Top + Y);
end;
```

See Also

Popup (see page 1434)

OnPopup (see page 1444)

PopupMenu

34.1.9.5.5 TPopupMenu.BiDiMode Property

Determines the layout of the menu in Middle East locales.

Pascal

```
property BiDiMode: TBiDiMode;
```

C++

```
__property TBiDiMode BiDiMode;
```

Description

Set BiDiMode to control the layout of menu items. BiDiMode controls whether the menu items read from left to right or right to left.

See Also

ParentBiDiMode (see page 1351)

IsRightToLeft (see page 1339)

34.1.9.5.6 TPopupMenu.HelpContext Property

Specifies a context ID for the entire pop-up menu.

Pascal

```
property HelpContext: THelpContext;
```

C++

```
__property THelpContext HelpContext;
```

Description

Use HelpContext to associate a help screen with the entire pop-up menu. Each screen in the Help system should have a unique context ID. When a component is selected in the application, pressing F1 displays the Help screen associated with the value of HelpContext.

To associate a help screen with the individual items in the pop-up menu, use the HelpContext property of the associated TMenuItem (see page 1363) objects.

See Also

GetHelpContext (see page 1338)

HelpFile

HelpContext (see page 1410)

34.1.9.5.7 TPopupMenu.Images Property

Lists the images that can appear beside individual menu items.

Pascal

```
property Images: TCustomImageList;
```

C++

```
__property TCustomImageList Images;
```

Description

Use Images to hold the set of images that can appear beside menu items in the menu. Individual menu items set their ImageIndex property to indicate which image in the list appears to the left of the menu item caption.

Images in the list should be 16x16.

Tip: If the application uses action lists, this property should be the value of the action list's Images property. This way, menu items and buttons that do the same thing can display the same image.

See Also

ImageIndex (see page 1411)

Images

Items (see page 1346)

34.1.9.5.8 TPopupMenu.MenuAnimation Property

Specifies how the menu appears when it is first displayed.

Pascal

```
property MenuAnimation: TMenuAnimation;
```

C++

```
__property TMenuAnimation MenuAnimation;
```

Description

Use MenuAnimation to specify how the popup menu appears on the screen. The menu can simply appear as a whole, or can

slide into view like a window shade being drawn from any edge.

Note: MenuAnimation only has an effect when running on Windows 98, NT 5.0, or later.

See Also

Popup (see page 1434)

34.1.9.5.9 TPopupMenu.OwnerDraw Property

Specifies whether the menu items are drawn by the application.

Pascal

```
property OwnerDraw: Boolean;
```

C++

```
__property Boolean OwnerDraw;
```

Description

Set OwnerDraw to true to indicate that the application will draw menu items in an OnDrawItem event handler. When OwnerDraw is true, menu items receive an OnMeasureItem and an OnDrawItem event when they need to be rendered on screen.

Set OwnerDraw to false to accept the default drawing. When OwnerDraw is false, menu items display the caption with an optional image to the left (If the Images and ImageIndex properties are set).

Owner-draw menus are useful for displaying the value that appears when selecting a menu item. For example, in a menu that allows users to choose a color, an owner-draw menu could display rectangles drawn in the designated color, rather than color names.

See Also

OnMeasureItem (see page 1423)

OnDrawItem (see page 1422)

Images (see page 1345)

34.1.9.5.10 TPopupMenu.ParentBiDiMode Property

Indicates whether the BiDiMode property is inherited from the control that uses the menu.

Pascal

```
property ParentBiDiMode: Boolean;
```

C++

```
__property Boolean ParentBiDiMode;
```

Description

By default, ParentBiDiMode is true, indicating that the layout of menu items follows the BiDiMode setting of the associated form or control. Change ParentBiDiMode to false to override the default BiDiMode setting and control the menu's layout independently.

See Also

BiDiMode (see page 1344)

34.1.9.5.11 TPopupMenu.PopupComponent Property

Indicates the component that last displayed the popup menu in response to a right mouse click.

Pascal

```
property PopupComponent: TComponent;
```

C++

```
__property TComponent * PopupComponent;
```

Description

Read PopupComponent to determine which control is currently using the popup menu. In applications where multiple controls share the same pop-up menu, use PopupComponent to determine which of them displayed the menu.

Set PopupComponent to associate a control with the menu before calling the Popup (see page 1434) method programmatically to bring up the pop-up menu.

C++ Examples:

```
/*
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  PopupMenu1->AutoPopup = true;
  Edit1->PopupMenu = PopupMenu1;
  Edit2->PopupMenu = PopupMenu1;
}

/*
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
*/
void __fastcall TForm1::Copy1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CopyToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Cut1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CutToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}
```

```
void __fastcall TForm1::Paste1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->PasteFromClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}
```

Delphi Examples:

```
{
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  PopupMenu1.AutoPopup := True;
  Edit1.PopupMenu := PopupMenu1;
  Edit2.PopupMenu := PopupMenu1;
end;

{
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
}
procedure TForm1.Copy1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
   Edit1.CopyToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CopyToClipboard
  else
    Beep;
end;

procedure TForm1.Cut1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.CutToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CutToClipboard
  else
    Beep;
end;

procedure TForm1.Paste1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.PasteFromClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.PasteFromClipboard
  else
    Beep;
```

```
end;
```

See Also

AutoPopup (see page 1437)

OnPopup (see page 1444)

Popup (see page 1434)

PopupMenu

34.1.9.5.12 TPopupMenu.PopupPoint Property

Indicates the location where the pop-up menu appears.

Pascal

```
property PopupPoint: TPoint;
```

C++

```
__property TPoint PopupPoint;
```

Description

This public property is used internally to locate the popup menu. Its value is set by the Popup (see page 1434) method.

See Also

Popup (see page 1434)

34.1.9.5.13 TPopupMenu.TrackButton Property

Specifies which mouse button activates the popup menu when it is associated with a toolbar button.

Pascal

```
property TrackButton: TTrackButton;
```

C++

```
__property TTrackButton TrackButton;
```

Description

Use TrackButton to specify which mouse button activates the popup menu when it is associated with a toolbar button.

34.1.9.6 TPopupMenu Events

The events of the TPopupMenu class are listed here.

Published Events

	Name	Description
	OnChange (see page 1443)	Occurs when the menu changes.
	OnPopup (see page 1444)	Occurs just before the pop-up menu appears.

34.1.9.6.1 TPopupMenu.OnChange Event

Occurs when the menu changes.

Pascal

```
property OnChange: TMenuChangeEvent;
```

C++

```
__property TMenuChangeEvent OnChange;
```

Description

Write an OnChange event handler to respond to changes in the menu's structure. OnChange occurs when the menu is loaded into memory, and in response to property changes that affect the menu's structure.

See Also

Items (see page 1346)

34.1.9.6.2 TPopupMenu.OnPopup Event

Occurs just before the pop-up menu appears.

Pascal

```
property OnPopup: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnPopup;
```

Description

Write an OnPopup event handler to take specific action just before the pop-up menu appears. For example, use an OnPopup event handler to set the Checked, Enabled, or Visible property of individual items in the menu so that they are appropriate to the PopupComponent (see page 1440).

C++ Examples:

```
/*
The following code enables the Copy item from the pop-up
menu if the edit control has a nonempty selection.
Note: A portion of the edit control text must be highlighted.
*/
void __fastcall TForm1::PopupMenu1Popup(TObject *Sender)
{
  Copy1->Enabled = (Edit1->SelLength > 0);
}

void __fastcall TForm1::Copy1Click(TObject *Sender)
{
    ListBox1->Items->Add("'Copy' menu");
}

void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
          int X, int Y)
{
  PopupMenu1->Popup(X, Y);
}
```

Delphi Examples:

```
{
The following code enables the Copy item from the pop-up
menu if the edit control has a nonempty selection.
Note: A portion of the edit control text must be highlighted.
}
procedure TForm1.PopupMenu1Popup(Sender: TObject);
begin
```

```
  Copy1.Enabled := Edit1.SelLength > 0;
end;

procedure TForm1.Copy1Click(Sender: TObject);
begin
    ListBox1.Items.Add('"Copy" menu');
end;

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  PopupMenu1.Popup(Left + X, Top + Y);
end;
```

See Also

AutoPopup (see page 1437)

Popup (see page 1434)

PopupComponent (see page 1440)

PopupMenu

34.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	AnsiSameCaption (see page 1446)	Indicates whether two strings are the same based on the current locale, ignoring case information and accelerator characters.
	DrawMenuItem (see page 1447)	This is function Menus.DrawMenuItem.
	GetHotkey (see page 1447)	Returns the last character in a string marked with an ampersand (&).
	IsAltGRPressed (see page 1447)	This is function Menus.IsAltGRPressed.
	NewItem (see page 1448)	Creates and initializes a menu item.
	NewLine (see page 1449)	Creates a menu item for a separator.
	NewMenu (see page 1450)	Creates and initializes a main menu.
	NewPopupMenu (see page 1451)	Creates and initializes a popup menu.
	NewSubMenu (see page 1452)	Creates and initializes a menu item that represents a submenu.
	SameCaption (see page 1453)	This is function Menus.SameCaption.
	ShortCut (see page 1453)	Creates a menu shortcut for a specified key combination.
	ShortCutFromMessage (see page 1454)	This is function Menus.ShortCutFromMessage.
	ShortCutToKey (see page 1454)	Returns the key code and shift state of a menu shortcut.
	ShortCutToText (see page 1456)	Converts a menu shortcut into a string that describes the shortcut.
	StripHotkey (see page 1457)	Removes the accelerator character marker from a string.
	TextToShortCut (see page 1457)	Creates a menu shortcut from a text string.

34.2.1 Menus.AnsiSameCaption Function

Indicates whether two strings are the same based on the current locale, ignoring case information and accelerator characters.

Pascal

```
function AnsiSameCaption(const Text1: string; const Text2: string): Boolean;
```

C++

```
Boolean AnsiSameCaption(const UnicodeString Text1, const UnicodeString Text2);
```

File

Menus

Description

AnsiSameCaption compares Text1 to Text2, without case sensitivity and ignoring any ampersands (&) that are included to create accelerator characters. The compare operation is controlled by the current locale. The return value is true if the two strings have the same value, false otherwise.

See Also

StringReplace

IsAccel

AnsiSameText

34.2.2 Menus.DrawMenuItem Function

Pascal

```
procedure DrawMenuItem(MenuItem: TMenuItem; ACanvas: TCanvas; ARect: TRect; State:
TOwnerDrawState);
```

C++

```
DrawMenuItem(TMenuItem MenuItem, TCanvas ACanvas, TRect ARect, TOwnerDrawState State);
```

File

Menus

Description

This is function Menus.DrawMenuItem.

34.2.3 Menus.GetHotkey Function

Returns the last character in a string marked with an ampersand (&).

Pascal

```
function GetHotkey(const Text: string): string;
```

C++

```
UnicodeString GetHotkey(const UnicodeString Text);
```

File

Menus

Description

GetHotKey searches Text, starting at the end, for a character that is identified as an accelerator key by a preceding ampersand (&). If it locates a character marked with an ampersand, GetHotkey returns that character. Otherwise, it returns an empty string.

See Also

AnsiSameCaption (see page 1446)

StripHotkey (see page 1457)

IsAccel

34.2.4 Menus.IsAltGRPressed Function

Pascal

```
function IsAltGRPressed: boolean;
```

C++

```
boolean IsAltGRPressed();
```

File

Menus

Description

This is function Menus.IsAltGRPressed.

34.2.5 Menus.NewItem Function

Creates and initializes a menu item.

Pascal

```
function NewItem(const ACaption: string; AShortCut: TShortCut; AChecked: Boolean; AEnabled:
Boolean; AOnClick: TNotifyEvent; hCtx: THelpContext; const AName: string): TMenuItem;
```

C++

```
TMenuItem NewItem(const UnicodeString ACaption, TShortCut AShortCut, Boolean AChecked, Boolean
AEnabled, TNotifyEvent AOnClick, THelpContext hCtx, const UnicodeString AName);
```

File

Menus

Description

Call NewItem to create and initialize a menu item when constructing a menu programmatically.

The ACaption parameter specifies the string that should appear for the menu item.

The AShortCut parameter specifies any associated shortcut.

The AChecked parameter indicates whether the menu item appears with a check mark.

The AEnabled parameter specifies whether the menu item is enabled or grayed.

The AOnClick parameter specifies an event handler that executes when the user clicks the menu item.

The hCtx parameter specifies the help context ID for the menu item.

The AName parameter specifies the name of the menu item, which can be used to refer to the menu item in code.

The menu item returned by NewItem can be added to a menu using the target parent's Add method. If the entire menu is being created dynamically at runtime, the new menu item can be added to an array that is passed as a parameter to the NewMenu (see page 1450), NewPopupMenu (see page 1451), or NewSubMenu (see page 1452) function.

Warning: The item returned by NewItem does not have an owner. You are responsible for freeing its memory when it is no longer needed. The Delete and Remove methods of TMenuItem (see page 1363) do not free memory.

C++ Examples:

```
/*
This code displays a new menu named MyMenu when the user
clicks the button. MyOnExecute must be included in the
definition of the TForm1 class.  Click on the menu item
or type "Ctrl+N" to execute MyOnExecute.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMenuItem *Item1 = NewItem("New item", TextToShortCut("Ctrl+N"), false, true, MyOnExecute,
```

```
0, "Item1");
  Menu = NewMenu(this, "MyMenu", &Item1, 0);
}

void __fastcall TForm1::MyOnExecute(TObject *Sender)
{
  MessageDlg("Item1 has been executed.", mtInformation, TMsgDlgButtons() << mbOK, 0);
}
```

Delphi Examples:

```
{
This code displays a new menu named MyMenu when the user
clicks the button. MyOnExecute must be included in the
definition of the TForm1 class.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Menu := NewMenu(Self, 'MyMenu',
            [NewItem('New item', TextToShortCut('Ctrl+N'),
            False, True, MyOnExecute, 0, 'Item1')]);
end;

procedure TForm1.MyOnExecute(Sender: TObject);
begin
  MessageDlg('Item1 has been executed.', mtInformation, [mbOK], 0);
end;
```

See Also

NewSubMenu (see page 1452)

NewLine (see page 1449)

NewMenu (see page 1450)

NewPopupMenu (see page 1451)

TMenuItem (see page 1363)

34.2.6 Menus.NewLine Function

Creates a menu item for a separator.

Pascal

```
function NewLine: TMenuItem;
```

C++

```
TMenuItem NewLine();
```

File

Menus

Description

Call NewLine to create a separator that divides the items in a menu into logical groups. Once the menu item for the separator has been created using NewLine, it can be added to a menu using the target parent's Add method. If the entire menu is being created dynamically at runtime, the new menu item can be added to an array that is passed as a parameter to the NewMenu (see page 1450), NewPopupMenu (see page 1451), or NewSubMenu (see page 1452) function.

Warning: The item returned by NewLine does not have an owner. You are responsible for freeing its memory when it is no

longer needed. The Delete and Remove methods of TMenuItem (see page 1363) do not free memory.

See Also

NewItem (see page 1448)

NewPopupMenu (see page 1451)

NewSubMenu (see page 1452)

TMenuItem (see page 1363)

34.2.7 Menus.NewMenu Function

Creates and initializes a main menu.

Pascal

```
function NewMenu(Owner: TComponent; const AName: string; const Items: array of TMenuItem):
TMainMenu;
```

C++

```
TMainMenu NewMenu(TComponent * Owner, const UnicodeString AName, const array of TMenuItem
Items);
```

File

Menus

Description

Call NewMenu to create and initialize a main menu programmatically. NewMenu returns the TMainMenu (see page 1322) component that represents the menu.

The Owner parameter specifies the component that is responsible for freeing the menu (typically the form).

The AName parameter specifies the name of the menu, which is used to refer to it in code.

The Items parameter is an array of menu items that make up the top level menu items. To create the menu items for the Items parameter, use the NewItem (see page 1448) function.

Note: The Items_Size parameter specifies the index of the last menu item in Items (one less than the number of menu items).

C++ Examples:

```
/*
This code displays a new menu named MyMenu when the user
clicks the button. MyOnExecute must be included in the
definition of the TForm1 class.  Click on the menu item
or type "Ctrl+N" to execute MyOnExecute.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMenuItem *Item1 = NewItem("New item", TextToShortCut("Ctrl+N"), false, true, MyOnExecute,
0, "Item1");
  Menu = NewMenu(this, "MyMenu", &Item1, 0);
}

void __fastcall TForm1::MyOnExecute(TObject *Sender)
{
  MessageDlg("Item1 has been executed.", mtInformation, TMsgDlgButtons() << mbOK, 0);
}
```

Delphi Examples:

```
{
This code displays a new menu named MyMenu when the user
clicks the button. MyOnExecute must be included in the
definition of the TForm1 class.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Menu := NewMenu(Self, 'MyMenu',
            [NewItem('New item', TextToShortCut('Ctrl+N'),
            False, True, MyOnExecute, 0, 'Item1')]);
end;

procedure TForm1.MyOnExecute(Sender: TObject);
begin
  MessageDlg('Item1 has been executed.', mtInformation, [mbOK], 0);
end;
```

See Also

NewPopupMenu (see page 1451)

NewSubMenu (see page 1452)

NewItem (see page 1448)

34.2.8 Menus.NewPopupMenu Function

Creates and initializes a popup menu.

Pascal

```
function NewPopupMenu(Owner: TComponent; const AName: string; Alignment: TPopupAlignment;
AutoPopup: Boolean; const Items: array of TMenuItem): TPopupMenu;
```

C++

```
TPopupMenu NewPopupMenu(TComponent * Owner, const UnicodeString AName, TPopupAlignment
Alignment, Boolean AutoPopup, const array of TMenuItem Items);
```

File

Menus

Description

Call NewPopupMenu to create and initialize a popup menu programmatically. NewPopupMenu returns the TPopupMenu (see page 1430) component that represents the popup menu, but does not associate that popup menu with any control.

The Owner parameter specifies the component that is responsible for freeing the menu.

The AName parameter specifies the name of the popup menu, which is used to refer to it in code.

The Alignment parameter determines where the menu appears relative to the mouse when the user right clicks on the control that uses the menu.

The AutoPopup parameter specifies whether the menu should appear automatically when the user right clicks the associated control.

The Items parameter is an array of menu items that make up the top level menu items. To create the menu items for the Items parameter, use the NewItem (see page 1448) function.

Note: The Items_Size parameter specifies the index of the last menu item in Items (one less than the number of menu items).

See Also

NewMenu (see page 1450)

NewSubMenu (see page 1452)

NewItem (see page 1448)

34.2.9 Menus.NewSubMenu Function

Creates and initializes a menu item that represents a submenu.

Pascal

```
function NewSubMenu(const ACaption: string; hCtx: THelpContext; const AName: string; const
Items: array of TMenuItem; AEnabled: Boolean = True): TMenuItem;
```

C++

```
TMenuItem NewSubMenu(const UnicodeString ACaption, THelpContext hCtx, const UnicodeString
AName, const array of TMenuItem Items, Boolean AEnabled = True);
```

File

Menus

Description

Call NewSubMenu to create a menu item that represents a submenu when constructing a menu programmatically at runtime.

The ACaption parameter indicates the string that appears in the parent menu for the submenu.

The hCtx parameter specifies the help context ID for the menu item.

The AName parameter specifies the name of the menu item, which can be used to refer to the menu item in code.

The Items parameter specifies an array of menu items that make up the submenu and that appear when the user clicks the submenu item in the parent menu.

Note: The Items_Size parameter specifies the index of the last menu item in Items (one less than the number of menu items).

The AEnabled parameter specifies whether the menu item that appears in the parent menu is enabled.

Once a submenu has been created using NewSubMenu, it can be added to a menu using the target parent's Add method. If the entire menu is being created dynamically at runtime, the new menu item can be added to an array that is passed as a parameter to the NewMenu (see page 1450), NewPopupMenu (see page 1451), or NewSubMenu function.

Warning: The item returned by NewSubMenu does not have an owner. You are responsible for freeing its memory when it is no longer needed. The Delete and Remove methods of TMenuItem (see page 1363) do not free memory.

See Also

NewItem (see page 1448)

TMenuItem (see page 1363)

NewLine (see page 1449)

NewMenu (see page 1450)

NewPopupMenu (see page 1451)

34.2.10 Menus.SameCaption Function

Pascal

```
function SameCaption(const Text1: string; const Text2: string): Boolean;
```

C++

```
Boolean SameCaption(const UnicodeString Text1, const UnicodeString Text2);
```

File

Menus

Description

This is function Menus.SameCaption.

34.2.11 Menus.ShortCut Function

Creates a menu shortcut for a specified key combination.

Pascal

```
function ShortCut(Key: Word; Shift: TShiftState): TShortCut;
```

C++

```
TShortCut ShortCut(Word Key, TShiftState Shift);
```

File

Menus

Description

Call ShortCut to create a menu shortcut at runtime.

Specify a Key value using a virtual key code (Menus (see page 1319) version) or Qt key code (QMenus version).

Specify the values for the Shift, Control, and Alt keys by using a set of type TShiftState. For example, to specify the Shift key, use the set [ssShift]. To specify a Shift and Ctrl key combination, use the set [ssShift, ssCtrl].

Assign the value returned by ShortCut as the value of the ShortCut property of a menu item.

Note: TextToShortCut (see page 1457) can also create a shortcut. TextToShortCut (see page 1457) converts a string to a shortcut, so it can be used to let the user specify the shortcut. However, TextToShortCut (see page 1457) executes much more slowly than ShortCut. Use ShortCut whenever possible.

C++ Examples:

```
/*
This example creates a shortcut, Ctrl+O, at run time and
assigns it to the Open command on a File menu.
*/
void __fastcall TForm1::OpenCommandClick(TObject *Sender)
{
  MessageDlg(
    "The Open command has been selected.",
```

```
    mtInformation,
    TMsgDlgButtons() << mbOK,
    0);
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  OpenCommand->ShortCut =
    ShortCut(Word('O'), TShiftState() << ssCtrl);
}
```

Delphi Examples:

```
{
This example creates a shortcut, Ctrl+O, at run time and
assigns it to the Open command on a File menu.
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  OpenCommand.ShortCut := Menus.ShortCut(Word('O'), [ssCtrl]);
end;

procedure TForm1.OpenCommandClick(Sender: TObject);
begin
  MessageDlg('The Open command has been selected.', mtInformation, [mbOk], 0);
end;
```

See Also

ShortCutToKey (see page 1454)

ShortCutToText (see page 1456)

TextToShortCut (see page 1457)

TMenuItem (see page 1363)

34.2.12 Menus.ShortCutFromMessage Function

Pascal

```
function ShortCutFromMessage(Message: TWMKey): TShortCut;
```

C++

```
TShortCut ShortCutFromMessage(TWMKey Message);
```

File

Menus

Description

This is function Menus.ShortCutFromMessage.

34.2.13 Menus.ShortCutToKey Function

Returns the key code and shift state of a menu shortcut.

Pascal

```
procedure ShortCutToKey(ShortCut: TShortCut; var Key: Word; var Shift: TShiftState);
```

C++

```
ShortCutToKey(TShortCut ShortCut, Word Key, TShiftState Shift);
```

File

Menus

Description

Call ShortCutToKey to parse a menu shortcut into its virtual key code and shift state parts.

ShortCut (see page 1453) is the value of the menu shortcut.

Key returns the virtual key code (Menus (see page 1319) version) or Qt key code (QMenus version) for the key.

Shirt returns the associated combination of Shift, Ctrl, and Alt keys.

C++ Examples:

```
/*
The following code redefines the ShortCut of CloseCommand if
the original short cut used the [ssCtrl] shift state.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Word TheKey;

  TShiftState TheShiftState;

  ShortCutToKey(CloseCommand->ShortCut, TheKey, TheShiftState);
  if (TheShiftState.Contains(ssCtrl))
    CloseCommand->ShortCut = ShortCut(Word('C'), TShiftState() << ssShift);}

void __fastcall TForm1::CloseCommandClick(TObject *Sender)
{
  MessageDlg(
    "The Close command has been selected.",
    mtInformation,
    TMsgDlgButtons() << mbOK,
    0);
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  CloseCommand->ShortCut =
    ShortCut(Word('C'), TShiftState() << ssCtrl);
}
```

Delphi Examples:

```
{
The following code redefines the ShortCut of CloseCommand if
the original short cut used the [ssCtrl] shift state.
}
procedure TForm1.Button1Click(Sender: TObject);
var
  TheKey: Word;
  TheShiftState: TShiftState;
begin
  Menus.ShortCutToKey(CloseCommand.ShortCut, TheKey, TheShiftState);
  if TheShiftState = [ssCtrl] then
   CloseCommand.ShortCut := ShortCut(Word('C'), [ssShift]);
end;
```

```
procedure TForm1.CloseCommandClick(Sender: TObject);
begin
  MessageDlg('The Close command has been selected.', mtInformation, [mbOk], 0);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  CloseCommand.ShortCut := Menus.ShortCut(Word('C'), [ssCtrl]);
end;
```

See Also

ShortCut (see page 1453)

ShortCutToText (see page 1456)

TextToShortCut (see page 1457)

TMenuItem (see page 1363)

34.2.14 Menus.ShortCutToText Function

Converts a menu shortcut into a string that describes the shortcut.

Pascal

```
function ShortCutToText(ShortCut: TShortCut): string;
```

C++

```
UnicodeString ShortCutToText(TShortCut ShortCut);
```

File

Menus

Description

Call ShortCutToText to obtain a text description of the shortcut associated with a menu item. Use this function when the application needs to display a description of a shortcut to the user.

C++ Examples:

```
/*
This line creates a shortcut, Ctrl+O, at run time and
assigns it to the Open command on a File menu.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Edit1->Text = ShortCutToText(OpenCommand->ShortCut);}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  OpenCommand->ShortCut =
    ShortCut(Word('O'), TShiftState() << ssCtrl);
}

void __fastcall TForm1::OpenCommandClick(TObject *Sender)
{
  MessageDlg(
    "The Open command has been selected.",
    mtInformation,
    TMsgDlgButtons() << mbOK,
    0);
}
```

Delphi Examples:

```
{
This line creates a shortcut, Ctrl+O, at run time and
assigns it to the Open command on a File menu.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Edit1.Text := Menus.ShortCutToText(OpenCommand.ShortCut);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  OpenCommand.ShortCut := Menus.ShortCut(Word('O'), [ssCtrl]);
end;

procedure TForm1.OpenCommandClick(Sender: TObject);
begin
  MessageDlg('The Open command has been selected.', mtInformation, [mbOk], 0);
end;
```

See Also

ShortCut (see page 1453)

ShortCutToKey (see page 1454)

TextToShortCut (see page 1457)

TMenuItem (see page 1363)

34.2.15 Menus.StripHotkey Function

Removes the accelerator character marker from a string.

Pascal

```
function StripHotkey(const Text: string): string;
```

C++

```
UnicodeString StripHotkey(const UnicodeString Text);
```

File

Menus

Description

StripHotkey removes the & escape char that marks the accelerator characters in Text. When the current locale is Asian, this function also looks for and removes any parentheses around the accelerator, which are common in Asian locales.

See Also

StringReplace

IsAccel

34.2.16 Menus.TextToShortCut Function

Creates a menu shortcut from a text string.

Pascal

```
function TextToShortCut(Text: string): TShortCut;
```

C++

```
TShortCut TextToShortCut(UnicodeString Text);
```

File

Menus

Description

Call TextToShortCut to create a menu shortcut from a descriptive string. For example, an application can allow users to specify the value of a shortcut in an edit box control, and then call TextToShortCut to create a menu shortcut for the value entered by the user.

Note: TextToShortCut executes slowly. When possible, use the ShortCut (see page 1453) function to create a menu shortcut instead.

C++ Examples:

```
/*
This code displays a new menu named MyMenu when the user
clicks the button. MyOnExecute must be included in the
definition of the TForm1 class.  Click on the menu item
or type "Ctrl+N" to execute MyOnExecute.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMenuItem *Item1 = NewItem("New item", TextToShortCut("Ctrl+N"), false, true, MyOnExecute,
0, "Item1");
  Menu = NewMenu(this, "MyMenu", &Item1, 0);
}

void __fastcall TForm1::MyOnExecute(TObject *Sender)
{
  MessageDlg("Item1 has been executed.", mtInformation, TMsgDlgButtons() << mbOK, 0);
}
```

Delphi Examples:

```
{
This code displays a new menu named MyMenu when the user
clicks the button. MyOnExecute must be included in the
definition of the TForm1 class.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Menu := NewMenu(Self, 'MyMenu',
            [NewItem('New item', TextToShortCut('Ctrl+N'),
            False, True, MyOnExecute, 0, 'Item1')]);
end;

procedure TForm1.MyOnExecute(Sender: TObject);
begin
  MessageDlg('Item1 has been executed.', mtInformation, [mbOK], 0);
end;
```

See Also

ShortCut (see page 1453)

ShortCutToText (see page 1456)

TMenuItem (see page 1363)

34.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TFindItemKind (see page 1460)	TFindItemKind indicates the criterion to use when locating menu items.
	TMenuAnimations (see page 1461)	TMenuAnimation (see page 1466) and TMenuAnimations indicate how a popup menu appears on screen.
	TMenuBreak (see page 1461)	TMenuBreak indicates the criterion to use when locating menu items.
	TMenuItemAutoFlag (see page 1462)	TMenuItemAutoFlag and TMenuAutoFlag (see page 1467) Indicate when menu items are automatically maintained.
	TPopupAlignment (see page 1463)	TPopupAlignment represents the position of a popup menu relative to the mouse.
	TTrackButton (see page 1463)	TTrackButton specifies which mouse button activates a tool button menu.

34.3.1 Menus.TFindItemKind Enumeration

TFindItemKind indicates the criterion to use when locating menu items.

Pascal

```
TFindItemKind = (
  fkCommand,
  fkHandle,
  fkShortCut
);
```

C++

```
enum TFindItemKind {
  fkCommand,
  fkHandle,
  fkShortCut
};
```

File

Menus

Description

TFindItemKind values indicate the criterion to use when locating a menu item. The following table gives the possible values:

Value	Meaning
fkID	Identify the item by its Windows command ID (parameter to WM_COMMAND (see page 1684) message).
fkHandle	Identify the item by its pop-up menu handle.
fkShortCut	Identify the item by its menu shortcut key code.

34.3.2 Menus.TMenuAnimations Enumeration

TMenuAnimation (see page 1466) and TMenuAnimations indicate how a popup menu appears on screen.

Pascal

```
TMenuAnimations = (
  maLeftToRight,
  maRightToLeft,
  maTopToBottom,
  maBottomToTop,
  maNone
);
```

C++

```
enum TMenuAnimations {
  maLeftToRight,
  maRightToLeft,
  maTopToBottom,
  maBottomToTop,
  maNone
};
```

File

Menus

Description

TMenuAnimations indicates the direction in which a menu can "slide" into view. TMenuAnimation (see page 1466) is a set of TMenuAnimations values, which can be used to specify more than one direction (for a popup menu that slides in at a diagonal). Note that some TMenuAnimations values are mutually exclusive (for example maLeftToRight and maRightToLeft).

Value	Meaning
maLeftToRight	The popup menu appears like a window shade that pulls from left to right.
maRightToLeft	The popup menu appears like a window shade that pulls from right to left.
maTopToBottom	The popup menu appears like a window shade that pulls from top to bottom.
maBottomToTop	The popup menu appears like a window shade that pulls from bottom to top.
maNone	The popup menu appears as a whole with no animation effects.

See Also

TPopupMenu (see page 1430)

34.3.3 Menus.TMenuBreak Enumeration

TMenuBreak indicates the criterion to use when locating menu items.

Pascal

```
TMenuBreak = (
  mbNone,
  mbBreak,
  mbBarBreak
```

```
);
```

C++

```
enum TMenuBreak {
  mbNone,
  mbBreak,
  mbBarBreak
};
```

File

Menus

Description

TMenuBreak values indicate the criterion to use when locating a menu item. The following table gives the possible values:

Value	Meaning
mbNone	No menu breaking occurs.
mbBarBreak	The Parent menu breaks into another column with the menu item appearing at the top of the new column. A bar separates the new and the old columns.
mbBreak	The Parent menu breaks into another column with the menu item appearing at the top of the new column. Only space separates the new and the old columns.

34.3.4 Menus.TMenuItemAutoFlag Enumeration

TMenuItemAutoFlag and TMenuAutoFlag (see page 1467) Indicate when menu items are automatically maintained.

Pascal

```
TMenuItemAutoFlag = (
  maAutomatic,
  maManual,
  maParent
);
```

C++

```
enum TMenuItemAutoFlag {
  maAutomatic,
  maManual,
  maParent
};
```

File

Menus

Description

TMenuItemAutoFlag indicates when a submenu is checked for problems and updated. TMenuAutoFlag (see page 1467) provides the same information for top-level menu items. The following table lists the possible values:

Values	Meaning
maAutomatic	The immediate submenu of the menu item or the top-level menu items of the menu are checked for problems and automatically corrected when the menu is displayed at runtime.
maManual	The immediate submenu of the menu item or the top-level menu items of the menu are checked for problems and automatically corrected only when the application specifically calls a method of the menu item to do so.

maParent	Automatic checking and correction of the immediate submenu follows the same rule as the menu item's parent. This value is not applicable to menus, which have no parent menu item.

See Also

AutoHotkeys (see page 1343)

AutoLineReduction (see page 1344)

34.3.5 Menus.TPopupAlignment Enumeration

TPopupAlignment represents the position of a popup menu relative to the mouse.

Pascal

```
TPopupAlignment = (
  paLeft,
  paRight,
  paCenter
);
```

C++

```
enum TPopupAlignment {
  paLeft,
  paRight,
  paCenter
};
```

File

Menus

Description

These are the possible values:

Value	Meaning
paLeft	The pop-up menu appears with its top left corner under the mouse.
paCenter	The pop-up menu appears with the top center of the menu under the mouse.
paRight	The pop-up menu appears with its top right corner under the mouse.

34.3.6 Menus.TTrackButton Enumeration

TTrackButton specifies which mouse button activates a tool button menu.

Pascal

```
TTrackButton = (
  tbRightButton,
  tbLeftButton
);
```

C++

```
enum TTrackButton {
```

```
    tbRightButton,
    tbLeftButton
};
```

File

Menus

Description

TTrackButton can have one of the following values:

Value	Meaning
tbRightButton	The right mouse button activates the tool button menu.
tbLeftButton	The left mouse button activates the tool button menu.

34.4 Types

The following table lists types in this documentation.

Types

Name	Description
PMenuItem (see page 1465)	This is type Menus.PMenuItem.
TAdvancedMenuDrawItemEvent (see page 1465)	TAdvancedMenuDrawItemEvent is the type for event handlers that render menu items in owner-drawn menus.
TMenuActionLinkClass (see page 1466)	This is type Menus.TMenuActionLinkClass.
TMenuAnimation (see page 1466)	TMenuAnimation and TMenuAnimations (see page 1461) indicate how a popup menu appears on screen.
TMenuAutoFlag (see page 1467)	TMenuItemAutoFlag (see page 1462) and TMenuAutoFlag Indicate when menu items are automatically maintained.
TMenuChangeEvent (see page 1467)	This is type Menus.TMenuChangeEvent.
TMenuDrawItemEvent (see page 1468)	TMenuDrawItemEvent is the type for event handlers that render menu items in owner-drawn menus.
TMenuHandle (see page 1468)	This is type Menus.TMenuHandle.
TMenuMeasureItemEvent (see page 1469)	TMenuMeasureItemEvent is the type for event handlers that render menu items in owner-Measuren menus.

34.4.1 Menus.PMenuItem Type

Pascal

```
PMenuItem = ^TMenuItem;
```

C++

```
^TMenuItem PMenuItem;
```

File

Menus

Description

This is type Menus.PMenuItem.

34.4.2 Menus.TAdvancedMenuDrawItemEvent Type

TAdvancedMenuDrawItemEvent is the type for event handlers that render menu items in owner-drawn menus.

Pascal

```
TAdvancedMenuDrawItemEvent = procedure (Sender: TObject; ACanvas: TCanvas; ARect: TRect;
State: TOwnerDrawState) of object;
```

C++

```
(Sender: TObject; ACanvas: TCanvas; ARect: TRect; State: TOwnerDrawState) (
TAdvancedMenuDrawItemEvent)();
```

File

Menus

Description

Sender is the TMenuItem (see page 1363) instance that represents the menu item to be drawn.

ACanvas provides a drawing surface on which to draw the menu item.

ARect indicates the boundaries (in pixels) of the menu item on ACanvas.

State is a set indicating various aspects of the menu items state such as whether it is selected, checked, grayed, and so on.

See Also

Procedural types

Creating events

34.4.3 Menus.TMenuActionLinkClass Type

Pascal

```
TMenuActionLinkClass = class of TMenuActionLink;
```

C++

```
TMenuActionLink TMenuActionLinkClass;
```

File

Menus

Description

This is type Menus.TMenuActionLinkClass.

34.4.4 Menus.TMenuAnimation Type

TMenuAnimation and TMenuAnimations (see page 1461) indicate how a popup menu appears on screen.

Pascal

```
TMenuAnimation = set of TMenuAnimations;
```

C++

```
TMenuAnimations TMenuAnimation;
```

File

Menus

Description

TMenuAnimations (see page 1461) indicates the direction in which a menu can "slide" into view. TMenuAnimation is a set of TMenuAnimations (see page 1461) values, which can be used to specify more than one direction (for a popup menu that slides in at a diagonal). Note that some TMenuAnimations (see page 1461) values are mutually exclusive (for example maLeftToRight and maRightToLeft).

Value	Meaning
maLeftToRight	The popup menu appears like a window shade that pulls from left to right.
maRightToLeft	The popup menu appears like a window shade that pulls from right to left.
maTopToBottom	The popup menu appears like a window shade that pulls from top to bottom.
maBottomToTop	The popup menu appears like a window shade that pulls from bottom to top.
maNone	The popup menu appears as a whole with no animation effects.

See Also

TPopupMenu (see page 1430)

34.4.5 Menus.TMenuAutoFlag Type

TMenuItemAutoFlag (see page 1462) and TMenuAutoFlag Indicate when menu items are automatically maintained.

Pascal

```
TMenuAutoFlag = maAutomatic..maManual;
```

C++

```
maAutomatic..maManual TMenuAutoFlag;
```

File

Menus

Description

TMenuItemAutoFlag (see page 1462) indicates when a submenu is checked for problems and updated. TMenuAutoFlag provides the same information for top-level menu items. The following table lists the possible values:

Values	Meaning
maAutomatic	The immediate submenu of the menu item or the top-level menu items of the menu are checked for problems and automatically corrected when the menu is displayed at runtime.
maManual	The immediate submenu of the menu item or the top-level menu items of the menu are checked for problems and automatically corrected only when the application specifically calls a method of the menu item to do so.
maParent	Automatic checking and correction of the immediate submenu follows the same rule as the menu item's parent. This value is not applicable to menus, which have no parent menu item.

See Also

AutoHotkeys (see page 1343)

AutoLineReduction (see page 1344)

34.4.6 Menus.TMenuChangeEvent Type

Pascal

```
TMenuChangeEvent = procedure (Sender: TObject; Source: TMenuItem; Rebuild: Boolean) of object;
```

C++

```
(Sender: TObject; Source: TMenuItem; Rebuild: Boolean) ( TMenuChangeEvent)();
```

File

Menus

Description

This is type Menus.TMenuChangeEvent.

34.4.7 Menus.TMenuDrawItemEvent Type

TMenuDrawItemEvent is the type for event handlers that render menu items in owner-drawn menus.

Pascal

```
TMenuDrawItemEvent = procedure (Sender: TObject; ACanvas: TCanvas; ARect: TRect; Selected:
Boolean) of object;
```

C++

```
(Sender: TObject; ACanvas: TCanvas; ARect: TRect; Selected: Boolean) ( TMenuDrawItemEvent)();
```

File

Menus

Description

Sender is the TMenuItem (see page 1363) instance that represents the menu item to be drawn.

ACanvas provides a drawing surface on which to draw the menu item.

ARect indicates the boundaries (in pixels) of the menu item on ACanvas.

Selected indicates whether the menu item is selected (that is, if it will be clicked if the user presses return)

See Also

Procedural types

Creating events

34.4.8 Menus.TMenuHandle Type

Pascal

```
TMenuHandle = HMENU;
```

C++

```
HMENU TMenuHandle;
```

File

Menus

Description

This is type Menus.TMenuHandle.

34.4.9 Menus.TMenuMeasureItemEvent Type

TMenuMeasureItemEvent is the type for event handlers that render menu items in owner-Measuren menus.

Pascal

```
TMenuMeasureItemEvent = procedure (Sender: TObject; ACanvas: TCanvas; var Width, Height:
Integer) of object;
```

C++

```
(Sender: TObject; ACanvas: TCanvas; var Width, Height: Integer) ( TMenuMeasureItemEvent)();
```

File

Menus

Description

Sender is the TMenuItem (see page 1363) instance that represents the menu item to be measured.

ACanvas represents the canvas where the menu item will be drawn. Use its methods, if necessary, to determine the size of the menu item.

Width and Height return the dimensions, in pixels, that should be used to draw the menu item.

See Also

Procedural types

Creating events

34.5 Variables

The following table lists variables in this documentation.

Variables

Name	Description
PopupList (see page 1470)	Provides centralized handling of Windows messages directed at popup menus.
ShortCutItems (see page 1470)	This is variable Menus.ShortCutItems.
ValidMenuHotkeys (see page 1471)	Determines what accelerator keys can be automatically assigned to a menu item caption.

34.5.1 Menus.PopupList Variable

Provides centralized handling of Windows messages directed at popup menus.

Pascal

```
PopupList: TPopupList;
```

C++

```
TPopupList PopupList;
```

File

Menus

Description

Use PopupList to access all the popup menus in the application or access the window handle of the window that responds to popup menu messages. PopupList maintains a list of all the popup menu components created in an application. It holds the handle of a hidden window that centralizes the processing of Windows messages directed at these popup menus.

See Also

TPopupMenu (see page 1430)

34.5.2 Menus.ShortCutItems Variable

Pascal

```
ShortCutItems: TMenuItemStack;
```

C++

```
TMenuItemStack ShortCutItems;
```

File

Menus

Description

This is variable Menus.ShortCutItems.

34.5.3 Menus.ValidMenuHotkeys Variable

Determines what accelerator keys can be automatically assigned to a menu item caption.

Pascal

```
ValidMenuHotkeys: string = '1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ';
```

C++

```
UnicodeString ValidMenuHotkeys = '1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ';
```

File

Menus

Description

ValidMenuHotkeys is used to limit the characters that a menu considers when automatically assigning accelerator keys to menu items. This variable is used at runtime when the AutoHotkeys property is true or when an application calls RethinkHotkeys.

By default, ValidMenuHotkeys limits accelerator keys to alphanumeric values (digits and letters). Change this variable to redefine the keys that can be used when assigning accelerator keys automatically.

See Also

AutoHotkeys (see page 1343)

TMenuItem (see page 1363)

34.6 Constants

The following table lists constants in this documentation.

Constants

Name	Description
cDialogSuffix (see page 1472)	This is constant Menus.cDialogSuffix.
cHotkeyPrefix (see page 1472)	This is constant Menus.cHotkeyPrefix.
cLineCaption (see page 1472)	This is constant Menus.cLineCaption.

34.6.1 Menus.cDialogSuffix Constant

Pascal

```
cDialogSuffix = '...';
```

C++

```
cDialogSuffix = '...';
```

File

Menus

Description

This is constant Menus.cDialogSuffix.

34.6.2 Menus.cHotkeyPrefix Constant

Pascal

```
cHotkeyPrefix = '&';
```

C++

```
cHotkeyPrefix = '&';
```

File

Menus

Description

This is constant Menus.cHotkeyPrefix.

34.6.3 Menus.cLineCaption Constant

Pascal

```
cLineCaption = '-';
```

C++

```
cLineCaption = '-';
```

File

Menus

Description

This is constant Menus.cLineCaption.

35 Messages Namespace

This is namespace Messages.

Topics

Name	Description
Functions (see page 1487)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 1511)	The following table lists structs, records, enums in this documentation.
Types (see page 1568)	The following table lists types in this documentation.
Constants (see page 1595)	The following table lists constants in this documentation.

Constants

Name	Description
BM_CLICK (see page 1602)	Represents the Windows message BM_CLICK.
BM_GETCHECK (see page 1603)	Represents the Windows message BM_GETCHECK.
BM_GETIMAGE (see page 1603)	Represents the Windows message BM_GETIMAGE.
BM_GETSTATE (see page 1604)	Represents the Windows message BM_GETSTATE.
BM_SETCHECK (see page 1604)	Represents the Windows message BM_SETCHECK.
BM_SETIMAGE (see page 1604)	Represents the Windows message BM_SETIMAGE.
BM_SETSTATE (see page 1605)	Represents the Windows message BM_SETSTATE.
BM_SETSTYLE (see page 1605)	Represents the Windows message BM_SETSTYLE.
BN_CLICKED (see page 1606)	Represents the Windows message BN_CLICKED.
BN_DBLCLK (see page 1606)	Represents the Windows message BN_DBLCLK.
BN_DISABLE (see page 1607)	Represents the Windows message BN_DISABLE.
BN_DOUBLECLICKED (see page 1607)	Represents the Windows message BN_DOUBLECLICKED.
BN_HILITE (see page 1607)	Represents the Windows message BN_HILITE.
BN_KILLFOCUS (see page 1608)	Represents the Windows message BN_KILLFOCUS.
BN_PAINT (see page 1608)	Represents the Windows message BN_PAINT.
BN_PUSHED (see page 1609)	Represents the Windows message BN_PUSHED.
BN_SETFOCUS (see page 1609)	Represents the Windows message BN_SETFOCUS.
BN_UNHILITE (see page 1610)	Represents the Windows message BN_UNHILITE.
BN_UNPUSHED (see page 1610)	Represents the Windows message BN_UNPUSHED.
CB_ADDSTRING (see page 1610)	Represents the Windows message CB_ADDSTRING.
CB_DELETESTRING (see page 1611)	Represents the Windows message CB_DELETESTRING.
CB_DIR (see page 1611)	Represents the Windows message CB_DIR.
CB_FINDSTRING (see page 1612)	Represents the Windows message CB_FINDSTRING.
CB_FINDSTRINGEXACT (see page 1612)	Represents the Windows message CB_FINDSTRINGEXACT.

CB_GETCOUNT (see page 1613)	Represents the Windows message CB_GETCOUNT.
CB_GETCURSEL (see page 1613)	Represents the Windows message CB_GETCURSEL.
CB_GETDROPPEDCONTROLRECT (see page 1613)	Represents the Windows message CB_GETDROPPEDCONTROLRECT.
CB_GETDROPPEDSTATE (see page 1614)	Represents the Windows message CB_GETDROPPEDSTATE.
CB_GETDROPPEDWIDTH (see page 1614)	Represents the Windows message CB_GETDROPPEDWIDTH.
CB_GETEDITSEL (see page 1615)	Represents the Windows message CB_GETEDITSEL.
CB_GETEXTENDEDUI (see page 1615)	Represents the Windows message CB_GETEXTENDEDUI.
CB_GETHORIZONTALEXTENT (see page 1616)	Represents the Windows message CB_GETHORIZONTALEXTENT.
CB_GETITEMDATA (see page 1616)	Represents the Windows message CB_GETITEMDATA.
CB_GETITEMHEIGHT (see page 1617)	Represents the Windows message CB_GETITEMHEIGHT.
CB_GETLBTEXT (see page 1617)	Represents the Windows message CB_GETLBTEXT.
CB_GETLBTEXTLEN (see page 1617)	Represents the Windows message CB_GETLBTEXTLEN.
CB_GETLOCALE (see page 1618)	Represents the Windows message CB_GETLOCALE.
CB_GETTOPINDEX (see page 1618)	Represents the Windows message CB_GETTOPINDEX.
CB_INITSTORAGE (see page 1619)	Represents the Windows message CB_INITSTORAGE.
CB_INSERTSTRING (see page 1619)	Represents the Windows message CB_INSERTSTRING.
CB_LIMITTEXT (see page 1620)	Represents the Windows message CB_LIMITTEXT.
CB_MSGMAX (see page 1620)	Represents the Windows message CB_MSGMAX.
CB_RESETCONTENT (see page 1620)	Represents the Windows message CB_RESETCONTENT.
CB_SELECTSTRING (see page 1621)	Represents the Windows message CB_SELECTSTRING.
CB_SETCURSEL (see page 1621)	Represents the Windows message CB_SETCURSEL.
CB_SETDROPPEDWIDTH (see page 1622)	Represents the Windows message CB_SETDROPPEDWIDTH.
CB_SETEDITSEL (see page 1622)	Represents the Windows message CB_SETEDITSEL.
CB_SETEXTENDEDUI (see page 1623)	Represents the Windows message CB_SETEXTENDEDUI.
CB_SETHORIZONTALEXTENT (see page 1623)	Represents the Windows message CB_SETHORIZONTALEXTENT.
CB_SETITEMDATA (see page 1623)	Represents the Windows message CB_SETITEMDATA.
CB_SETITEMHEIGHT (see page 1624)	Represents the Windows message CB_SETITEMHEIGHT.
CB_SETLOCALE (see page 1624)	Represents the Windows message CB_SETLOCALE.
CB_SETTOPINDEX (see page 1625)	Represents the Windows message CB_SETTOPINDEX.
CB_SHOWDROPDOWN (see page 1625)	Represents the Windows message CB_SHOWDROPDOWN.
CBN_CLOSEUP (see page 1626)	Represents the Windows message CBN_CLOSEUP.
CBN_DBLCLK (see page 1626)	Represents the Windows message CBN_DBLCLK.
CBN_DROPDOWN (see page 1626)	Represents the Windows message CBN_DROPDOWN.
CBN_EDITCHANGE (see page 1627)	Represents the Windows message CBN_EDITCHANGE.
CBN_EDITUPDATE (see page 1627)	Represents the Windows message CBN_EDITUPDATE.
CBN_ERRSPACE (see page 1628)	Represents the Windows message CBN_ERRSPACE.
CBN_KILLFOCUS (see page 1628)	Represents the Windows message CBN_KILLFOCUS.
CBN_SELCHANGE (see page 1629)	Represents the Windows message CBN_SELCHANGE.
CBN_SELENDCANCEL (see page 1629)	Represents the Windows message CBN_SELENDCANCEL.
CBN_SELENDOK (see page 1629)	Represents the Windows message CBN_SELENDOK.
CBN_SETFOCUS (see page 1630)	Represents the Windows message CBN_SETFOCUS.
DM_GETDEFID (see page 1630)	Represents the Windows message DM_GETDEFID.
DM_REPOSITION (see page 1631)	Represents the Windows message DM_REPOSITION.
DM_SETDEFID (see page 1631)	Represents the Windows message DM_SETDEFID.
EM_CANUNDO (see page 1632)	Represents the Windows message EM_CANUNDO.
EM_CHARFROMPOS (see page 1632)	Represents the Windows message EM_CHARFROMPOS.
EM_EMPTYUNDOBUFFER (see page 1632)	Represents the Windows message EM_EMPTYUNDOBUFFER.
EM_FMTLINES (see page 1633)	Represents the Windows message EM_FMTLINES.
EM_GETFIRSTVISIBLELINE (see page 1633)	Represents the Windows message EM_GETFIRSTVISIBLELINE.
EM_GETHANDLE (see page 1634)	Represents the Windows message EM_GETHANDLE.
EM_GETIMESTATUS (see page 1634)	Represents the Windows message EM_GETIMESTATUS.
EM_GETLIMITTEXT (see page 1635)	Represents the Windows message EM_GETLIMITTEXT.
EM_GETLINE (see page 1635)	Represents the Windows message EM_GETLINE.
EM_GETLINECOUNT (see page 1635)	Represents the Windows message EM_GETLINECOUNT.

EM_GETMARGINS (see page 1636)	Represents the Windows message EM_GETMARGINS.
EM_GETMODIFY (see page 1636)	Represents the Windows message EM_GETMODIFY.
EM_GETPASSWORDCHAR (see page 1637)	Represents the Windows message EM_GETPASSWORDCHAR.
EM_GETRECT (see page 1637)	Represents the Windows message EM_GETRECT.
EM_GETSEL (see page 1638)	Represents the Windows message EM_GETSEL.
EM_GETTHUMB (see page 1638)	Represents the Windows message EM_GETTHUMB.
EM_GETWORDBREAKPROC (see page 1638)	Represents the Windows message EM_GETWORDBREAKPROC.
EM_LIMITTEXT (see page 1639)	Represents the Windows message EM_LIMITTEXT.
EM_LINEFROMCHAR (see page 1639)	Represents the Windows message EM_LINEFROMCHAR.
EM_LINEINDEX (see page 1640)	Represents the Windows message EM_LINEINDEX.
EM_LINELENGTH (see page 1640)	Represents the Windows message EM_LINELENGTH.
EM_LINESCROLL (see page 1641)	Represents the Windows message EM_LINESCROLL.
EM_POSFROMCHAR (see page 1641)	Represents the Windows message EM_POSFROMCHAR.
EM_REPLACESEL (see page 1641)	Represents the Windows message EM_REPLACESEL.
EM_SCROLL (see page 1642)	Represents the Windows message EM_SCROLL.
EM_SCROLLCARET (see page 1642)	Represents the Windows message EM_SCROLLCARET.
EM_SETHANDLE (see page 1643)	Represents the Windows message EM_SETHANDLE.
EM_SETIMESTATUS (see page 1643)	Represents the Windows message EM_SETIMESTATUS.
EM_SETLIMITTEXT (see page 1644)	Represents the Windows message EM_SETLIMITTEXT.
EM_SETMARGINS (see page 1644)	Represents the Windows message EM_SETMARGINS.
EM_SETMODIFY (see page 1644)	Represents the Windows message EM_SETMODIFY.
EM_SETPASSWORDCHAR (see page 1645)	Represents the Windows message EM_SETPASSWORDCHAR.
EM_SETREADONLY (see page 1645)	Represents the Windows message EM_SETREADONLY.
EM_SETRECT (see page 1646)	Represents the Windows message EM_SETRECT.
EM_SETRECTNP (see page 1646)	Represents the Windows message EM_SETRECTNP.
EM_SETSEL (see page 1647)	Represents the Windows message EM_SETSEL.
EM_SETTABSTOPS (see page 1647)	Represents the Windows message EM_SETTABSTOPS.
EM_SETWORDBREAKPROC (see page 1647)	Represents the Windows message EM_SETWORDBREAKPROC.
EM_UNDO (see page 1648)	Represents the Windows message EM_UNDO.
EN_CHANGE (see page 1648)	Represents the Windows message EN_CHANGE.
EN_ERRSPACE (see page 1649)	Represents the Windows message EN_ERRSPACE.
EN_HSCROLL (see page 1649)	Represents the Windows message EN_HSCROLL.
EN_KILLFOCUS (see page 1650)	Represents the Windows message EN_KILLFOCUS.
EN_MAXTEXT (see page 1650)	Represents the Windows message EN_MAXTEXT.
EN_SETFOCUS (see page 1650)	Represents the Windows message EN_SETFOCUS.
EN_UPDATE (see page 1651)	Represents the Windows message EN_UPDATE.
EN_VSCROLL (see page 1651)	Represents the Windows message EN_VSCROLL.
LB_ADDFILE (see page 1652)	Represents the Windows message LB_ADDFILE.
LB_ADDSTRING (see page 1652)	Represents the Windows message LB_ADDSTRING.
LB_DELETESTRING (see page 1653)	Represents the Windows message LB_DELETESTRING.
LB_DIR (see page 1653)	Represents the Windows message LB_DIR.
LB_FINDSTRING (see page 1653)	Represents the Windows message LB_FINDSTRING.
LB_FINDSTRINGEXACT (see page 1654)	Represents the Windows message LB_FINDSTRINGEXACT.
LB_GETANCHORINDEX (see page 1654)	Represents the Windows message LB_GETANCHORINDEX.
LB_GETCARETINDEX (see page 1655)	Represents the Windows message LB_GETCARETINDEX.
LB_GETCOUNT (see page 1655)	Represents the Windows message LB_GETCOUNT.
LB_GETCURSEL (see page 1656)	Represents the Windows message LB_GETCURSEL.
LB_GETHORIZONTALEXTENT (see page 1656)	Represents the Windows message LB_GETHORIZONTALEXTENT.
LB_GETITEMDATA (see page 1656)	Represents the Windows message LB_GETITEMDATA.
LB_GETITEMHEIGHT (see page 1657)	Represents the Windows message LB_GETITEMHEIGHT.
LB_GETITEMRECT (see page 1657)	Represents the Windows message LB_GETITEMRECT.
LB_GETLOCALE (see page 1658)	Represents the Windows message LB_GETLOCALE.
LB_GETSEL (see page 1658)	Represents the Windows message LB_GETSEL.
LB_GETSELCOUNT (see page 1659)	Represents the Windows message LB_GETSELCOUNT.

LB_GETSELITEMS (see page 1659)	Represents the Windows message LB_GETSELITEMS.
LB_GETTEXT (see page 1659)	Represents the Windows message LB_GETTEXT.
LB_GETTEXTLEN (see page 1660)	Represents the Windows message LB_GETTEXTLEN.
LB_GETTOPINDEX (see page 1660)	Represents the Windows message LB_GETTOPINDEX.
LB_INITSTORAGE (see page 1661)	Represents the Windows message LB_INITSTORAGE.
LB_INSERTSTRING (see page 1661)	Represents the Windows message LB_INSERTSTRING.
LB_ITEMFROMPOINT (see page 1662)	Represents the Windows message LB_ITEMFROMPOINT.
LB_MSGMAX (see page 1662)	Represents the Windows message LB_MSGMAX.
LB_RESETCONTENT (see page 1662)	Represents the Windows message LB_RESETCONTENT.
LB_SELECTSTRING (see page 1663)	Represents the Windows message LB_SELECTSTRING.
LB_SELITEMRANGE (see page 1663)	Represents the Windows message LB_SELITEMRANGE.
LB_SELITEMRANGEEX (see page 1664)	Represents the Windows message LB_SELITEMRANGEEX.
LB_SETANCHORINDEX (see page 1664)	Represents the Windows message LB_SETANCHORINDEX.
LB_SETCARETINDEX (see page 1665)	Represents the Windows message LB_SETCARETINDEX.
LB_SETCOLUMNWIDTH (see page 1665)	Represents the Windows message LB_SETCOLUMNWIDTH.
LB_SETCOUNT (see page 1665)	Represents the Windows message LB_SETCOUNT.
LB_SETCURSEL (see page 1666)	Represents the Windows message LB_SETCURSEL.
LB_SETHORIZONTALEXTENT (see page 1666)	Represents the Windows message LB_SETHORIZONTALEXTENT.
LB_SETITEMDATA (see page 1667)	Represents the Windows message LB_SETITEMDATA.
LB_SETITEMHEIGHT (see page 1667)	Represents the Windows message LB_SETITEMHEIGHT.
LB_SETLOCALE (see page 1668)	Represents the Windows message LB_SETLOCALE.
LB_SETSEL (see page 1668)	Represents the Windows message LB_SETSEL.
LB_SETTABSTOPS (see page 1668)	Represents the Windows message LB_SETTABSTOPS.
LB_SETTOPINDEX (see page 1669)	Represents the Windows message LB_SETTOPINDEX.
LBN_DBLCLK (see page 1669)	Represents the Windows message LBN_DBLCLK.
LBN_ERRSPACE (see page 1670)	Represents the Windows message LBN_ERRSPACE.
LBN_KILLFOCUS (see page 1670)	Represents the Windows message LBN_KILLFOCUS.
LBN_SELCANCEL (see page 1671)	Represents the Windows message LBN_SELCANCEL.
LBN_SELCHANGE (see page 1671)	Represents the Windows message LBN_SELCHANGE.
LBN_SETFOCUS (see page 1671)	Represents the Windows message LBN_SETFOCUS.
PSM_PAGEINFO (see page 1672)	Represents the Windows message PSM_PAGEINFO.
PSM_SHEETINFO (see page 1672)	Represents the Windows message PSM_SHEETINFO.
SBM_ENABLE_ARROWS (see page 1673)	Represents the Windows message SBM_ENABLE_ARROWS.
SBM_GETPOS (see page 1673)	Represents the Windows message SBM_GETPOS.
SBM_GETRANGE (see page 1674)	Represents the Windows message SBM_GETRANGE.
SBM_GETSCROLLBARINFO (see page 1674)	Represents the Windows message SBM_GETSCROLLBARINFO.
SBM_GETSCROLLINFO (see page 1674)	Represents the Windows message SBM_GETSCROLLINFO.
SBM_SETPOS (see page 1675)	Represents the Windows message SBM_SETPOS.
SBM_SETRANGE (see page 1675)	Represents the Windows message SBM_SETRANGE.
SBM_SETRANGEREDRAW (see page 1676)	Represents the Windows message SBM_SETRANGEREDRAW.
SBM_SETSCROLLINFO (see page 1676)	Represents the Windows message SBM_SETSCROLLINFO.
WM_ACTIVATE (see page 1677)	Represents the Windows message WM_ACTIVATE.
WM_ACTIVATEAPP (see page 1677)	Represents the Windows message WM_ACTIVATEAPP.
WM_APP (see page 1678)	Represents the Windows message WM_APP.
WM_APPCOMMAND (see page 1678)	Represents the Windows message WM_APPCOMMAND.
WM_ASKCBFORMATNAME (see page 1678)	Represents the Windows message WM_ASKCBFORMATNAME.
WM_CANCELJOURNAL (see page 1679)	Represents the Windows message WM_CANCELJOURNAL.
WM_CANCELMODE (see page 1679)	Represents the Windows message WM_CANCELMODE.
WM_CAPTURECHANGED (see page 1680)	Represents the Windows message WM_CAPTURECHANGED.
WM_CHANGECBCHAIN (see page 1680)	Represents the Windows message WM_CHANGECBCHAIN.
WM_CHANGEUISTATE (see page 1681)	Represents the Windows message WM_CHANGEUISTATE.
WM_CHAR (see page 1681)	Represents the Windows message WM_CHAR.
WM_CHARTOITEM (see page 1681)	Represents the Windows message WM_CHARTOITEM.
WM_CHILDACTIVATE (see page 1682)	Represents the Windows message WM_CHILDACTIVATE.

WM_CLEAR (see page 1682)	Represents the Windows message WM_CLEAR.
WM_CLOSE (see page 1683)	Represents the Windows message WM_CLOSE.
WM_COALESCE_FIRST (see page 1683)	Represents the Windows message WM_COALESCE_FIRST.
WM_COALESCE_LAST (see page 1684)	Represents the Windows message WM_COALESCE_LAST.
WM_COMMAND (see page 1684)	Represents the Windows message WM_COMMAND.
WM_COMMNOTIFY (see page 1685)	Represents the Windows message WM_COMMNOTIFY.
WM_COMPACTING (see page 1685)	Represents the Windows message WM_COMPACTING.
WM_COMPAREITEM (see page 1685)	Represents the Windows message WM_COMPAREITEM.
WM_CONTEXTMENU (see page 1686)	Represents the Windows message WM_CONTEXTMENU.
WM_COPY (see page 1686)	Represents the Windows message WM_COPY.
WM_COPYDATA (see page 1687)	Represents the Windows message WM_COPYDATA.
WM_CREATE (see page 1687)	Represents the Windows message WM_CREATE.
WM_CTLCOLOR (see page 1688)	Represents the Windows message WM_CTLCOLOR.
WM_CTLCOLORBTN (see page 1688)	Represents the Windows message WM_CTLCOLORBTN.
WM_CTLCOLORDLG (see page 1688)	Represents the Windows message WM_CTLCOLORDLG.
WM_CTLCOLOREDIT (see page 1689)	Represents the Windows message WM_CTLCOLOREDIT.
WM_CTLCOLORLISTBOX (see page 1689)	Represents the Windows message WM_CTLCOLORLISTBOX.
WM_CTLCOLORMSGBOX (see page 1690)	Represents the Windows message WM_CTLCOLORMSGBOX.
WM_CTLCOLORSCROLLBAR (see page 1690)	Represents the Windows message WM_CTLCOLORSCROLLBAR.
WM_CTLCOLORSTATIC (see page 1691)	Represents the Windows message WM_CTLCOLORSTATIC.
WM_CUT (see page 1691)	Represents the Windows message WM_CUT.
WM_DDE_ACK (see page 1691)	Represents the Windows message WM_DDE_ACK.
WM_DDE_ADVISE (see page 1692)	Represents the Windows message WM_DDE_ADVISE.
WM_DDE_DATA (see page 1692)	Represents the Windows message WM_DDE_DATA.
WM_DDE_EXECUTE (see page 1693)	Represents the Windows message WM_DDE_EXECUTE.
WM_DDE_FIRST (see page 1693)	Represents the Windows message WM_DDE_FIRST.
WM_DDE_INITIATE (see page 1694)	Represents the Windows message WM_DDE_INITIATE.
WM_DDE_LAST (see page 1694)	Represents the Windows message WM_DDE_LAST.
WM_DDE_POKE (see page 1695)	Represents the Windows message WM_DDE_POKE.
WM_DDE_REQUEST (see page 1695)	Represents the Windows message WM_DDE_REQUEST.
WM_DDE_TERMINATE (see page 1695)	Represents the Windows message WM_DDE_TERMINATE.
WM_DDE_UNADVISE (see page 1696)	Represents the Windows message WM_DDE_UNADVISE.
WM_DEADCHAR (see page 1696)	Represents the Windows message WM_DEADCHAR.
WM_DELETEITEM (see page 1697)	Represents the Windows message WM_DELETEITEM.
WM_DESTROY (see page 1697)	Represents the Windows message WM_DESTROY.
WM_DESTROYCLIPBOARD (see page 1698)	Represents the Windows message WM_DESTROYCLIPBOARD.
WM_DEVICECHANGE (see page 1698)	Represents the Windows message WM_DEVICECHANGE.
WM_DEVMODECHANGE (see page 1698)	Represents the Windows message WM_DEVMODECHANGE.
WM_DISPLAYCHANGE (see page 1699)	Represents the Windows message WM_DISPLAYCHANGE.
WM_DRAWCLIPBOARD (see page 1699)	Represents the Windows message WM_DRAWCLIPBOARD.
WM_DRAWITEM (see page 1700)	Represents the Windows message WM_DRAWITEM.
WM_DROPFILES (see page 1700)	Represents the Windows message WM_DROPFILES.
WM_DWMCOLORIZATIONCOLORCHANGED (see page 1701)	Represents the Windows message WM_DWMCOLORIZATIONCOLORCHANGED.
WM_DWMCOMPOSITIONCHANGED (see page 1701)	Represents the Windows message WM_DWMCOMPOSITIONCHANGED.
WM_DWMNCRENDERINGCHANGED (see page 1702)	Represents the Windows message WM_DWMNCRENDERINGCHANGED.
WM_DWMWINDOWMAXIMIZEDCHANGE (see page 1702)	Represents the Windows message WM_DWMWINDOWMAXIMIZEDCHANGE.
WM_ENABLE (see page 1703)	Represents the Windows message WM_ENABLE.
WM_ENDSESSION (see page 1703)	Represents the Windows message WM_ENDSESSION.
WM_ENTERIDLE (see page 1703)	Represents the Windows message WM_ENTERIDLE.
WM_ENTERMENULOOP (see page 1704)	Represents the Windows message WM_ENTERMENULOOP.
WM_ENTERSIZEMOVE (see page 1704)	Represents the Windows message WM_ENTERSIZEMOVE.
WM_ERASEBKGND (see page 1705)	Represents the Windows message WM_ERASEBKGND.
WM_EXITMENULOOP (see page 1705)	Represents the Windows message WM_EXITMENULOOP.
WM_EXITSIZEMOVE (see page 1706)	Represents the Windows message WM_EXITSIZEMOVE.

WM_FONTCHANGE (see page 1706)	Represents the Windows message WM_FONTCHANGE.
WM_GETDLGCODE (see page 1707)	Represents the Windows message WM_GETDLGCODE.
WM_GETFONT (see page 1707)	Represents the Windows message WM_GETFONT.
WM_GETHOTKEY (see page 1707)	Represents the Windows message WM_GETHOTKEY.
WM_GETICON (see page 1708)	Represents the Windows message WM_GETICON.
WM_GETMINMAXINFO (see page 1708)	Represents the Windows message WM_GETMINMAXINFO.
WM_GETOBJECT (see page 1709)	Represents the Windows message WM_GETOBJECT.
WM_GETTEXT (see page 1709)	Represents the Windows message WM_GETTEXT.
WM_GETTEXTLENGTH (see page 1710)	Represents the Windows message WM_GETTEXTLENGTH.
WM_HANDHELDFIRST (see page 1710)	Represents the Windows message WM_HANDHELDFIRST.
WM_HANDHELDLAST (see page 1711)	Represents the Windows message WM_HANDHELDLAST.
WM_HELP (see page 1711)	Represents the Windows message WM_HELP.
WM_HOTKEY (see page 1711)	Represents the Windows message WM_HOTKEY.
WM_HSCROLL (see page 1712)	Represents the Windows message WM_HSCROLL.
WM_HSCROLLCLIPBOARD (see page 1712)	Represents the Windows message WM_HSCROLLCLIPBOARD.
WM_ICONERASEBKGND (see page 1713)	Represents the Windows message WM_ICONERASEBKGND.
WM_IME_CHAR (see page 1713)	Represents the Windows message WM_IME_CHAR.
WM_IME_COMPOSITION (see page 1714)	Represents the Windows message WM_IME_COMPOSITION.
WM_IME_COMPOSITIONFULL (see page 1714)	Represents the Windows message WM_IME_COMPOSITIONFULL.
WM_IME_CONTROL (see page 1715)	Represents the Windows message WM_IME_CONTROL.
WM_IME_ENDCOMPOSITION (see page 1715)	Represents the Windows message WM_IME_ENDCOMPOSITION.
WM_IME_KEYDOWN (see page 1715)	Represents the Windows message WM_IME_KEYDOWN.
WM_IME_KEYLAST (see page 1716)	Represents the Windows message WM_IME_KEYLAST.
WM_IME_KEYUP (see page 1716)	Represents the Windows message WM_IME_KEYUP.
WM_IME_NOTIFY (see page 1717)	Represents the Windows message WM_IME_NOTIFY.
WM_IME_REQUEST (see page 1717)	Represents the Windows message WM_IME_REQUEST.
WM_IME_SELECT (see page 1718)	Represents the Windows message WM_IME_SELECT.
WM_IME_SETCONTEXT (see page 1718)	Represents the Windows message WM_IME_SETCONTEXT.
WM_IME_STARTCOMPOSITION (see page 1718)	Represents the Windows message WM_IME_STARTCOMPOSITION.
WM_INITDIALOG (see page 1719)	Represents the Windows message WM_INITDIALOG.
WM_INITMENU (see page 1719)	Represents the Windows message WM_INITMENU.
WM_INITMENUPOPUP (see page 1720)	Represents the Windows message WM_INITMENUPOPUP.
WM_INPUT (see page 1720)	Represents the Windows message WM_INPUT.
WM_INPUTLANGCHANGE (see page 1721)	Represents the Windows message WM_INPUTLANGCHANGE.
WM_INPUTLANGCHANGEREQUEST (see page 1721)	Represents the Windows message WM_INPUTLANGCHANGEREQUEST.
WM_KEYDOWN (see page 1722)	Represents the Windows message WM_KEYDOWN.
WM_KEYFIRST (see page 1722)	Represents the Windows message WM_KEYFIRST.
WM_KEYLAST (see page 1722)	Represents the Windows message WM_KEYLAST.
WM_KEYUP (see page 1723)	Represents the Windows message WM_KEYUP.
WM_KILLFOCUS (see page 1723)	Represents the Windows message WM_KILLFOCUS.
WM_LBUTTONDBLCLK (see page 1724)	Represents the Windows message WM_LBUTTONDBLCLK.
WM_LBUTTONDOWN (see page 1724)	Represents the Windows message WM_LBUTTONDOWN.
WM_LBUTTONUP (see page 1725)	Represents the Windows message WM_LBUTTONUP.
WM_MBUTTONDBLCLK (see page 1725)	Represents the Windows message WM_MBUTTONDBLCLK.
WM_MBUTTONDOWN (see page 1726)	Represents the Windows message WM_MBUTTONDOWN.
WM_MBUTTONUP (see page 1726)	Represents the Windows message WM_MBUTTONUP.
WM_MDIACTIVATE (see page 1726)	Represents the Windows message WM_MDIACTIVATE.
WM_MDICASCADE (see page 1727)	Represents the Windows message WM_MDICASCADE.
WM_MDICREATE (see page 1727)	Represents the Windows message WM_MDICREATE.
WM_MDIDESTROY (see page 1728)	Represents the Windows message WM_MDIDESTROY.
WM_MDIGETACTIVE (see page 1728)	Represents the Windows message WM_MDIGETACTIVE.
WM_MDIICONARRANGE (see page 1729)	Represents the Windows message WM_MDIICONARRANGE.
WM_MDIMAXIMIZE (see page 1729)	Represents the Windows message WM_MDIMAXIMIZE.
WM_MDINEXT (see page 1729)	Represents the Windows message WM_MDINEXT.

WM_MDIREFRESHMENU (see page 1730)	Represents the Windows message WM_MDIREFRESHMENU.
WM_MDIRESTORE (see page 1730)	Represents the Windows message WM_MDIRESTORE.
WM_MDISETMENU (see page 1731)	Represents the Windows message WM_MDISETMENU.
WM_MDITILE (see page 1731)	Represents the Windows message WM_MDITILE.
WM_MEASUREITEM (see page 1732)	Represents the Windows message WM_MEASUREITEM.
WM_MENUCHAR (see page 1732)	Represents the Windows message WM_MENUCHAR.
WM_MENUCOMMAND (see page 1732)	Represents the Windows message WM_MENUCOMMAND.
WM_MENUDRAG (see page 1733)	Represents the Windows message WM_MENUDRAG.
WM_MENUGETOBJECT (see page 1733)	Represents the Windows message WM_MENUGETOBJECT.
WM_MENURBUTTONUP (see page 1734)	Represents the Windows message WM_MENURBUTTONUP.
WM_MENUSELECT (see page 1734)	Represents the Windows message WM_MENUSELECT.
WM_MOUSEACTIVATE (see page 1735)	Represents the Windows message WM_MOUSEACTIVATE.
WM_MOUSEFIRST (see page 1735)	Represents the Windows message WM_MOUSEFIRST.
WM_MOUSEHOVER (see page 1736)	Represents the Windows message WM_MOUSEHOVER.
WM_MOUSELAST (see page 1736)	Represents the Windows message WM_MOUSELAST.
WM_MOUSELEAVE (see page 1736)	Represents the Windows message WM_MOUSELEAVE.
WM_MOUSEMOVE (see page 1737)	Represents the Windows message WM_MOUSEMOVE.
WM_MOUSEWHEEL (see page 1737)	Represents the Windows message WM_MOUSEWHEEL.
WM_MOVE (see page 1738)	Represents the Windows message WM_MOVE.
WM_MOVING (see page 1738)	Represents the Windows message WM_MOVING.
WM_NCACTIVATE (see page 1739)	Represents the Windows message WM_NCACTIVATE.
WM_NCCALCSIZE (see page 1739)	Represents the Windows message WM_NCCALCSIZE.
WM_NCCREATE (see page 1739)	Represents the Windows message WM_NCCREATE.
WM_NCDESTROY (see page 1740)	Represents the Windows message WM_NCDESTROY.
WM_NCHITTEST (see page 1740)	Represents the Windows message WM_NCHITTEST.
WM_NCLBUTTONDBLCLK (see page 1741)	Represents the Windows message WM_NCLBUTTONDBLCLK.
WM_NCLBUTTONDOWN (see page 1741)	Represents the Windows message WM_NCLBUTTONDOWN.
WM_NCLBUTTONUP (see page 1742)	Represents the Windows message WM_NCLBUTTONUP.
WM_NCMBUTTONDBLCLK (see page 1742)	Represents the Windows message WM_NCMBUTTONDBLCLK.
WM_NCMBUTTONDOWN (see page 1742)	Represents the Windows message WM_NCMBUTTONDOWN.
WM_NCMBUTTONUP (see page 1743)	Represents the Windows message WM_NCMBUTTONUP.
WM_NCMOUSEHOVER (see page 1743)	Represents the Windows message WM_NCMOUSEHOVER.
WM_NCMOUSELEAVE (see page 1744)	Represents the Windows message WM_NCMOUSELEAVE.
WM_NCMOUSEMOVE (see page 1744)	Represents the Windows message WM_NCMOUSEMOVE.
WM_NCPAINT (see page 1745)	Represents the Windows message WM_NCPAINT.
WM_NCRBUTTONDBLCLK (see page 1745)	Represents the Windows message WM_NCRBUTTONDBLCLK.
WM_NCRBUTTONDOWN (see page 1746)	Represents the Windows message WM_NCRBUTTONDOWN.
WM_NCRBUTTONUP (see page 1746)	Represents the Windows message WM_NCRBUTTONUP.
WM_NCXBUTTONDBLCLK (see page 1746)	Represents the Windows message WM_NCXBUTTONDBLCLK.
WM_NCXBUTTONDOWN (see page 1747)	Represents the Windows message WM_NCXBUTTONDOWN.
WM_NCXBUTTONUP (see page 1747)	Represents the Windows message WM_NCXBUTTONUP.
WM_NEXTDLGCTL (see page 1748)	Represents the Windows message WM_NEXTDLGCTL.
WM_NEXTMENU (see page 1748)	Represents the Windows message WM_NEXTMENU.
WM_NOTIFY (see page 1749)	Represents the Windows message WM_NOTIFY.
WM_NOTIFYFORMAT (see page 1749)	Represents the Windows message WM_NOTIFYFORMAT.
WM_NULL (see page 1749)	Represents the Windows message WM_NULL.
WM_PAINT (see page 1750)	Represents the Windows message WM_PAINT.
WM_PAINTCLIPBOARD (see page 1750)	Represents the Windows message WM_PAINTCLIPBOARD.
WM_PAINTICON (see page 1751)	Represents the Windows message WM_PAINTICON.
WM_PALETTECHANGED (see page 1751)	Represents the Windows message WM_PALETTECHANGED.
WM_PALETTEISCHANGING (see page 1752)	Represents the Windows message WM_PALETTEISCHANGING.
WM_PARENTNOTIFY (see page 1752)	Represents the Windows message WM_PARENTNOTIFY.
WM_PASTE (see page 1752)	Represents the Windows message WM_PASTE.
WM_PENWINFIRST (see page 1753)	Represents the Windows message WM_PENWINFIRST.

WM_PENWINLAST (see page 1753)	Represents the Windows message WM_PENWINLAST.
WM_POWER (see page 1754)	Represents the Windows message WM_POWER.
WM_POWERBROADCAST (see page 1754)	Represents the Windows message WM_POWERBROADCAST.
WM_PRINT (see page 1755)	Represents the Windows message WM_PRINT.
WM_PRINTCLIENT (see page 1755)	Represents the Windows message WM_PRINTCLIENT.
WM_QUERYDRAGICON (see page 1756)	Represents the Windows message WM_QUERYDRAGICON.
WM_QUERYENDSESSION (see page 1756)	Represents the Windows message WM_QUERYENDSESSION.
WM_QUERYNEWPALETTE (see page 1756)	Represents the Windows message WM_QUERYNEWPALETTE.
WM_QUERYOPEN (see page 1757)	Represents the Windows message WM_QUERYOPEN.
WM_QUERYUISTATE (see page 1757)	Represents the Windows message WM_QUERYUISTATE.
WM_QUEUESYNC (see page 1758)	Represents the Windows message WM_QUEUESYNC.
WM_QUIT (see page 1758)	Represents the Windows message WM_QUIT.
WM_RBUTTONDBLCLK (see page 1759)	Represents the Windows message WM_RBUTTONDBLCLK.
WM_RBUTTONDOWN (see page 1759)	Represents the Windows message WM_RBUTTONDOWN.
WM_RBUTTONUP (see page 1759)	Represents the Windows message WM_RBUTTONUP.
WM_RENDERALLFORMATS (see page 1760)	Represents the Windows message WM_RENDERALLFORMATS.
WM_RENDERFORMAT (see page 1760)	Represents the Windows message WM_RENDERFORMAT.
WM_SETCURSOR (see page 1761)	Represents the Windows message WM_SETCURSOR.
WM_SETFOCUS (see page 1761)	Represents the Windows message WM_SETFOCUS.
WM_SETFONT (see page 1762)	Represents the Windows message WM_SETFONT.
WM_SETHOTKEY (see page 1762)	Represents the Windows message WM_SETHOTKEY.
WM_SETICON (see page 1763)	Represents the Windows message WM_SETICON.
WM_SETREDRAW (see page 1763)	Represents the Windows message WM_SETREDRAW.
WM_SETTEXT (see page 1763)	Represents the Windows message WM_SETTEXT.
WM_SETTINGCHANGE (see page 1764)	Represents the Windows message WM_SETTINGCHANGE.
WM_SHOWWINDOW (see page 1764)	Represents the Windows message WM_SHOWWINDOW.
WM_SIZE (see page 1765)	Represents the Windows message WM_SIZE.
WM_SIZECLIPBOARD (see page 1765)	Represents the Windows message WM_SIZECLIPBOARD.
WM_SIZING (see page 1766)	Represents the Windows message WM_SIZING.
WM_SPOOLERSTATUS (see page 1766)	Represents the Windows message WM_SPOOLERSTATUS.
WM_STYLECHANGED (see page 1766)	Represents the Windows message WM_STYLECHANGED.
WM_STYLECHANGING (see page 1767)	Represents the Windows message WM_STYLECHANGING.
WM_SYSCHAR (see page 1767)	Represents the Windows message WM_SYSCHAR.
WM_SYSCOLORCHANGE (see page 1768)	Represents the Windows message WM_SYSCOLORCHANGE.
WM_SYSCOMMAND (see page 1768)	Represents the Windows message WM_SYSCOMMAND.
WM_SYSDEADCHAR (see page 1769)	Represents the Windows message WM_SYSDEADCHAR.
WM_SYSKEYDOWN (see page 1769)	Represents the Windows message WM_SYSKEYDOWN.
WM_SYSKEYUP (see page 1770)	Represents the Windows message WM_SYSKEYUP.
WM_SYSTEMERROR (see page 1770)	Represents the Windows message WM_SYSTEMERROR.
WM_TABLET_FIRST (see page 1770)	Represents the Windows message WM_TABLET_FIRST.
WM_TABLET_LAST (see page 1771)	Represents the Windows message WM_TABLET_LAST.
WM_TCARD (see page 1771)	Represents the Windows message WM_TCARD.
WM_THEMECHANGED (see page 1772)	Represents the Windows message WM_THEMECHANGED.
WM_TIMECHANGE (see page 1772)	Represents the Windows message WM_TIMECHANGE.
WM_TIMER (see page 1773)	Represents the Windows message WM_TIMER.
WM_UNDO (see page 1773)	Represents the Windows message WM_UNDO.
WM_UNICHAR (see page 1774)	Represents the Windows message WM_UNICHAR.
WM_UNINITMENUPOPUP (see page 1774)	Represents the Windows message WM_UNINITMENUPOPUP.
WM_UPDATEUISTATE (see page 1774)	Represents the Windows message WM_UPDATEUISTATE.
WM_USER (see page 1775)	Represents the Windows message WM_USER.
WM_USERCHANGED (see page 1775)	Represents the Windows message WM_USERCHANGED.
WM_VKEYTOITEM (see page 1776)	Represents the Windows message WM_VKEYTOITEM.
WM_VSCROLL (see page 1776)	Represents the Windows message WM_VSCROLL.
WM_VSCROLLCLIPBOARD (see page 1777)	Represents the Windows message WM_VSCROLLCLIPBOARD.

WM_WINDOWPOSCHANGED (see page 1777)	Represents the Windows message WM_WINDOWPOSCHANGED.
WM_WINDOWPOSCHANGING (see page 1778)	Represents the Windows message WM_WINDOWPOSCHANGING.
WM_WININICHANGE (see page 1778)	Represents the Windows message WM_WININICHANGE.
WM_WTSSESSION_CHANGE (see page 1778)	Represents the Windows message WM_WTSSESSION_CHANGE.

Functions

	Name	Description
	SendGetIntMessage (see page 1487)	Send Windows message, sending and receiving integer values.
	SendGetStructMessage (see page 1493)	Send Windows message, sending and receiving data structure.
	SendStructMessage (see page 1499)	Send Windows message, sending data structure.
	SendTextMessage (see page 1505)	Send Windows message, sending text.

Structs, Records, Enums

	Name	Description
	TMessage (see page 1513)	TMessage represents a Windows message.
	TMSHMouseWheel (see page 1514)	This is record Messages.TMSHMouseWheel.
	TWMActivate (see page 1514)	This is record Messages.TWMActivate.
	TWMActivateApp (see page 1515)	This is record Messages.TWMActivateApp.
	TWMAskCBFormatName (see page 1515)	This is record Messages.TWMAskCBFormatName.
	TWMChangeCBChain (see page 1516)	This is record Messages.TWMChangeCBChain.
	TWMCharToItem (see page 1516)	This is record Messages.TWMCharToItem.
	TWMChooseFont_GetLogFont (see page 1517)	This is record Messages.TWMChooseFont_GetLogFont.
	TWMCommand (see page 1517)	This is record Messages.TWMCommand.
	TWMCompacting (see page 1518)	This is record Messages.TWMCompacting.
	TWMCompareItem (see page 1518)	This is record Messages.TWMCompareItem.
	TWMContextMenu (see page 1519)	This is record Messages.TWMContextMenu.
	TWMCopyData (see page 1519)	This is record Messages.TWMCopyData.
	TWMCreate (see page 1520)	This is record Messages.TWMCreate.
	TWMCtlColor (see page 1520)	This is record Messages.TWMCtlColor.
	TWMDDE_Ack (see page 1521)	This is record Messages.TWMDDE_Ack.
	TWMDDE_Advise (see page 1521)	This is record Messages.TWMDDE_Advise.
	TWMDDE_Data (see page 1522)	This is record Messages.TWMDDE_Data.
	TWMDDE_Execute (see page 1522)	This is record Messages.TWMDDE_Execute.
	TWMDDE_Initiate (see page 1523)	This is record Messages.TWMDDE_Initiate.
	TWMDDE_Poke (see page 1523)	This is record Messages.TWMDDE_Poke.
	TWMDDE_Request (see page 1524)	This is record Messages.TWMDDE_Request.
	TWMDDE_Terminate (see page 1524)	This is record Messages.TWMDDE_Terminate.
	TWMDDE_Unadvise (see page 1525)	This is record Messages.TWMDDE_Unadvise.
	TWMDeleteItem (see page 1525)	This is record Messages.TWMDeleteItem.
	TWMDevModeChange (see page 1526)	This is record Messages.TWMDevModeChange.
	TWMDisplayChange (see page 1526)	This is record Messages.TWMDisplayChange.
	TWMDrawItem (see page 1527)	This is record Messages.TWMDrawItem.
	TWMDropFiles (see page 1527)	This is record Messages.TWMDropFiles.
	TWMEnable (see page 1528)	This is record Messages.TWMEnable.
	TWMEndSession (see page 1528)	This is record Messages.TWMEndSession.
	TWMEnterIdle (see page 1529)	This is record Messages.TWMEnterIdle.
	TWMEnterMenuLoop (see page 1529)	This is record Messages.TWMEnterMenuLoop.
	TWMEraseBkgnd (see page 1530)	This is record Messages.TWMEraseBkgnd.
	TWMGetIcon (see page 1530)	This is record Messages.TWMGetIcon.
	TWMGetMinMaxInfo (see page 1531)	This is record Messages.TWMGetMinMaxInfo.
	TWMGetText (see page 1531)	This is record Messages.TWMGetText.
	TWMHelp (see page 1532)	This is record Messages.TWMHelp.
	TWMHotKey (see page 1532)	This is record Messages.TWMHotKey.
	TWMHScrollClipboard (see page 1533)	This is record Messages.TWMHScrollClipboard.
	TWMInitDialog (see page 1533)	This is record Messages.TWMInitDialog.

	TWMInitMenu (see page 1534)	This is record Messages.TWMInitMenu.
	TWMInitMenuPopup (see page 1534)	This is record Messages.TWMInitMenuPopup.
	TWMKey (see page 1535)	TWMKey represents a Windows Keystroke message.
	TWMKillFocus (see page 1536)	This is record Messages.TWMKillFocus.
	TWMMDIActivate (see page 1536)	This is record Messages.TWMMDIActivate.
	TWMMDICascade (see page 1537)	This is record Messages.TWMMDICascade.
	TWMMDICreate (see page 1537)	This is record Messages.TWMMDICreate.
	TWMMDIDestroy (see page 1538)	This is record Messages.TWMMDIDestroy.
	TWMMDIMaximize (see page 1538)	This is record Messages.TWMMDIMaximize.
	TWMMDINext (see page 1539)	This is record Messages.TWMMDINext.
	TWMMDIRestore (see page 1539)	This is record Messages.TWMMDIRestore.
	TWMMDISetMenu (see page 1540)	This is record Messages.TWMMDISetMenu.
	TWMMDITile (see page 1540)	This is record Messages.TWMMDITile.
	TWMMeasureItem (see page 1541)	This is record Messages.TWMMeasureItem.
	TWMMenuChar (see page 1541)	TWMMenuChar represents a Windows Keystroke message directed at a menu.
	TWMMenuSelect (see page 1542)	This is record Messages.TWMMenuSelect.
	TWMMouse (see page 1543)	TWMMouse represents a Windows mouse message.
	TWMMouseActivate (see page 1544)	This is record Messages.TWMMouseActivate.
	TWMMouseWheel (see page 1544)	This is record Messages.TWMMouseWheel.
	TWMMove (see page 1545)	This is record Messages.TWMMove.
	TWMMoving (see page 1545)	This is record Messages.TWMMoving.
	TWMNCActivate (see page 1546)	This is record Messages.TWMNCActivate.
	TWMNCCalcSize (see page 1546)	This is record Messages.TWMNCCalcSize.
	TWMNCCreate (see page 1547)	This is record Messages.TWMNCCreate.
	TWMNCHitMessage (see page 1547)	This is record Messages.TWMNCHitMessage.
	TWMNCHitTest (see page 1548)	This is record Messages.TWMNCHitTest.
	TWMNCPaint (see page 1548)	This is record Messages.TWMNCPaint.
	TWMNextDlgCtl (see page 1549)	This is record Messages.TWMNextDlgCtl.
	TWMNoParams (see page 1549)	This is record Messages.TWMNoParams.
	TWMNotify (see page 1550)	TWMNotify represents a Windows notification message.
	TWMNotifyFormat (see page 1550)	This is record Messages.TWMNotifyFormat.
	TWMPaint (see page 1551)	TWMPaint represents a Windows Paint message.
	TWMPaintClipboard (see page 1551)	This is record Messages.TWMPaintClipboard.
	TWMPaletteChanged (see page 1552)	This is record Messages.TWMPaletteChanged.
	TWMPaletteIsChanging (see page 1552)	This is record Messages.TWMPaletteIsChanging.
	TWMParentNotify (see page 1553)	This is record Messages.TWMParentNotify.
	TWMPower (see page 1554)	This is record Messages.TWMPower.
	TWMPrint (see page 1554)	This is record Messages.TWMPrint.
	TWMQueryEndSession (see page 1555)	This is record Messages.TWMQueryEndSession.
	TWMQuit (see page 1555)	This is record Messages.TWMQuit.
	TWMRenderFormat (see page 1556)	This is record Messages.TWMRenderFormat.
	TWMScroll (see page 1556)	This is record Messages.TWMScroll.
	TWMSetCursor (see page 1557)	This is record Messages.TWMSetCursor.
	TWMSetFocus (see page 1557)	This is record Messages.TWMSetFocus.
	TWMSetFont (see page 1558)	This is record Messages.TWMSetFont.
	TWMSetHotKey (see page 1558)	This is record Messages.TWMSetHotKey.
	TWMSetIcon (see page 1559)	This is record Messages.TWMSetIcon.
	TWMSetRedraw (see page 1559)	This is record Messages.TWMSetRedraw.
	TWMSetText (see page 1560)	This is record Messages.TWMSetText.
	TWMSettingChange (see page 1560)	This is record Messages.TWMSettingChange.
	TWMShowWindow (see page 1561)	This is record Messages.TWMShowWindow.
	TWMSize (see page 1561)	This is record Messages.TWMSize.
	TWMSizeClipboard (see page 1562)	This is record Messages.TWMSizeClipboard.
	TWMSpoolerStatus (see page 1562)	This is record Messages.TWMSpoolerStatus.

	TWMStyleChange (see page 1563)	This is record Messages.TWMStyleChange.
	TWMSysCommand (see page 1563)	This is record Messages.TWMSysCommand.
	TWMSysDeadChar (see page 1564)	This is record Messages.TWMSysDeadChar.
	TWMSystemError (see page 1564)	This is record Messages.TWMSystemError.
	TWMTimer (see page 1565)	This is record Messages.TWMTimer.
	TWMUIState (see page 1565)	This is record Messages.TWMUIState.
	TWMVScrollClipboard (see page 1566)	This is record Messages.TWMVScrollClipboard.
	TWMWindowPosMsg (see page 1566)	This is record Messages.TWMWindowPosMsg.
	TWMWinIniChange (see page 1567)	This is record Messages.TWMWinIniChange.

Types

Name	Description
PMessage (see page 1569)	This is type Messages.PMessage.
TWMCancelMode (see page 1570)	This is type Messages.TWMCancelMode.
TWMChangeUIState (see page 1570)	This is type Messages.TWMChangeUIState.
TWMChar (see page 1570)	This is type Messages.TWMChar.
TWMChildActivate (see page 1571)	This is type Messages.TWMChildActivate.
TWMClear (see page 1571)	This is type Messages.TWMClear.
TWMClose (see page 1571)	This is type Messages.TWMClose.
TWMCopy (see page 1572)	This is type Messages.TWMCopy.
TWMCtlColorBtn (see page 1572)	This is type Messages.TWMCtlColorBtn.
TWMCtlColorDlg (see page 1572)	This is type Messages.TWMCtlColorDlg.
TWMCtlColorEdit (see page 1573)	This is type Messages.TWMCtlColorEdit.
TWMCtlColorListbox (see page 1573)	This is type Messages.TWMCtlColorListbox.
TWMCtlColorMsgbox (see page 1573)	This is type Messages.TWMCtlColorMsgbox.
TWMCtlColorScrollbar (see page 1574)	This is type Messages.TWMCtlColorScrollbar.
TWMCtlColorStatic (see page 1574)	This is type Messages.TWMCtlColorStatic.
TWMCut (see page 1574)	This is type Messages.TWMCut.
TWMDeadChar (see page 1575)	This is type Messages.TWMDeadChar.
TWMDestroy (see page 1575)	This is type Messages.TWMDestroy.
TWMDestroyClipboard (see page 1575)	This is type Messages.TWMDestroyClipboard.
TWMDrawClipboard (see page 1576)	This is type Messages.TWMDrawClipboard.
TWMExitMenuLoop (see page 1576)	This is type Messages.TWMExitMenuLoop.
TWMFontChange (see page 1576)	This is type Messages.TWMFontChange.
TWMGetDlgCode (see page 1577)	This is type Messages.TWMGetDlgCode.
TWMGetFont (see page 1577)	This is type Messages.TWMGetFont.
TWMGetHotKey (see page 1577)	This is type Messages.TWMGetHotKey.
TWMGetTextLength (see page 1578)	This is type Messages.TWMGetTextLength.
TWMHScroll (see page 1578)	This is type Messages.TWMHScroll.
TWMIconEraseBkgnd (see page 1578)	This is type Messages.TWMIconEraseBkgnd.
TWMKeyDown (see page 1579)	This is type Messages.TWMKeyDown.
TWMKeyUp (see page 1579)	This is type Messages.TWMKeyUp.
TWMLButtonDblClk (see page 1579)	This is type Messages.TWMLButtonDblClk.
TWMLButtonDown (see page 1580)	TWMLButtonDown represents the Windows message that occurs when the user clicks the left mouse button.
TWMLButtonUp (see page 1580)	This is type Messages.TWMLButtonUp.
TWMMButtonDblClk (see page 1580)	This is type Messages.TWMMButtonDblClk.
TWMMButtonDown (see page 1581)	This is type Messages.TWMMButtonDown.
TWMMButtonUp (see page 1581)	This is type Messages.TWMMButtonUp.
TWMMDIGetActive (see page 1581)	This is type Messages.TWMMDIGetActive.
TWMMDIIconArrange (see page 1582)	This is type Messages.TWMMDIIconArrange.
TWMMDIRefreshMenu (see page 1582)	This is type Messages.TWMMDIRefreshMenu.
TWMMouseMove (see page 1582)	This is type Messages.TWMMouseMove.
TWMNCDestroy (see page 1583)	This is type Messages.TWMNCDestroy.
TWMNCLButtonDblClk (see page 1583)	This is type Messages.TWMNCLButtonDblClk.

TWMNCLButtonDown (see page 1583)	This is type Messages.TWMNCLButtonDown.
TWMNCLButtonUp (see page 1584)	This is type Messages.TWMNCLButtonUp.
TWMNCMButtonDblClk (see page 1584)	This is type Messages.TWMNCMButtonDblClk.
TWMNCMButtonDown (see page 1584)	This is type Messages.TWMNCMButtonDown.
TWMNCMButtonUp (see page 1585)	This is type Messages.TWMNCMButtonUp.
TWMNCMouseMove (see page 1585)	This is type Messages.TWMNCMouseMove.
TWMNCRButtonDblClk (see page 1585)	This is type Messages.TWMNCRButtonDblClk.
TWMNCRButtonDown (see page 1586)	This is type Messages.TWMNCRButtonDown.
TWMNCRButtonUp (see page 1586)	This is type Messages.TWMNCRButtonUp.
TWMPaintIcon (see page 1586)	This is type Messages.TWMPaintIcon.
TWMPaste (see page 1587)	This is type Messages.TWMPaste.
TWMPrintClient (see page 1587)	This is type Messages.TWMPrintClient.
TWMQueryDragIcon (see page 1587)	This is type Messages.TWMQueryDragIcon.
TWMQueryNewPalette (see page 1588)	This is type Messages.TWMQueryNewPalette.
TWMQueryOpen (see page 1588)	This is type Messages.TWMQueryOpen.
TWMQueryUIState (see page 1588)	This is type Messages.TWMQueryUIState.
TWMQueueSync (see page 1589)	This is type Messages.TWMQueueSync.
TWMRButtonDblClk (see page 1589)	This is type Messages.TWMRButtonDblClk.
TWMRButtonDown (see page 1589)	This is type Messages.TWMRButtonDown.
TWMRButtonUp (see page 1590)	This is type Messages.TWMRButtonUp.
TWMRenderAllFormats (see page 1590)	This is type Messages.TWMRenderAllFormats.
TWMStyleChanged (see page 1590)	This is type Messages.TWMStyleChanged.
TWMStyleChanging (see page 1591)	This is type Messages.TWMStyleChanging.
TWMSysChar (see page 1591)	This is type Messages.TWMSysChar.
TWMSysColorChange (see page 1591)	This is type Messages.TWMSysColorChange.
TWMSysKeyDown (see page 1592)	This is type Messages.TWMSysKeyDown.
TWMSysKeyUp (see page 1592)	This is type Messages.TWMSysKeyUp.
TWMTimeChange (see page 1592)	This is type Messages.TWMTimeChange.
TWMUndo (see page 1593)	This is type Messages.TWMUndo.
TWMUpdateUIState (see page 1593)	This is type Messages.TWMUpdateUIState.
TWMVKeyToItem (see page 1593)	This is type Messages.TWMVKeyToItem.
TWMVScroll (see page 1594)	This is type Messages.TWMVScroll.
TWMWindowPosChanged (see page 1594)	This is type Messages.TWMWindowPosChanged.
TWMWindowPosChanging (see page 1594)	This is type Messages.TWMWindowPosChanging.

35.1 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	SendGetIntMessage (see page 1487)	Send Windows message, sending and receiving integer values.
	SendGetStructMessage (see page 1493)	Send Windows message, sending and receiving data structure.
	SendStructMessage (see page 1499)	Send Windows message, sending data structure.
	SendTextMessage (see page 1505)	Send Windows message, sending text.

35.1.1 Messages.SendGetIntMessage Function

Send Windows message, sending and receiving integer values.

Pascal

```
function SendGetIntMessage(Handle: HWND; Msg: UINT; var WParam: Integer; var LParam: Integer):
LRESULT;
```

C++

```
LRESULT SendGetIntMessage(HWND Handle, UINT Msg, int WParam, int LParam);
```

File

Messages

Description

SendGetIntMessage calls the Windows **SendMessage** function and returns an **LRESULT**.

Handle is an **HWND** window handle.

Msg is the Windows message ID. This constant is exactly the same as the Windows message ID described in the Microsoft Windows SDK.

WParam is a **WPARAM** for the message, depending on **Msg**. A value may be sent and returned.

LParam is an integer **LPARAM** for the message, depending on **Msg**. A value may be sent and returned.

Various records have been defined in the **Messages (see page 1475)** unit to represent Windows messages. For instance, **TWMCopy (see page 1572)** represents the message **WM_COPY (see page 1686)**, **TWMKeyDown (see page 1579)** represents **WM_KEYDOWN (see page 1722)**, and so on.

See the Microsoft Windows SDK documentation for information on the Windows messages and their associated **WPARAM** and **LPARAM**.

C++ Examples:

```
/*
This is an example demonstrating the usage of the SendGetIntMessage,
SendStructMessage, SendGetStructMessage and SendTextMessage message
functions.
This example uses four buttons and six text edit boxes as follows:
- Button1 //SendGetIntMessage
```

```
- Button2 //SendStructMessage
- Button3 //SendGetStructMessage
- Button4 //SendTextMessage

- IntEdit //for int inputs
- StructEdit1 & StructEdit2 //for struct inputs (ints)
- GetStructEdit1 & GetStructEdit2 //for struct inputs (ints)
- TextEdit //for text input

Note: This example generates unresolved external errors with the
default NO_STRICT conditional define set.  The work around is
to use the STRICT conditional define.  This problem is addressed
in the RAID bug #267447.

This example uses three custom message structures defined in the header:
struct TMyIntMessage {
    Cardinal Msg;
    int* wparam;
    int* lparam;
    LRESULT Result;
};

struct TMyTextMessage {
    Cardinal Msg;
    int wparam;
    String lparam;
    LRESULT Result;
};

struct MStruct {
    int m_a;
    int m_b;
};

struct TMyStruct {
    Cardinal Msg;
    int wparam;
    MStruct* lparam;
    LRESULT Result;
};

Furthermore it uses 4 message handling functions described in the
header along with a message map:

MESSAGE void __fastcall UserIntHandler(TMyIntMessage &msg);
MESSAGE void __fastcall UserStructHandler(TMyStruct &msg);
MESSAGE void __fastcall UserGetStructHandler(TMyStruct &msg);
MESSAGE void __fastcall UserTextHandler(TMyTextMessage &msg);

BEGIN_MESSAGE_MAP
    MESSAGE_HANDLER(WM_INTMESSAGE, TMyIntMessage, UserIntHandler);
    MESSAGE_HANDLER(WM_GETSTRUCTMESSAGE, TMyStruct, UserGetStructHandler);
    MESSAGE_HANDLER(WM_STRUCTMESSAGE, TMyStruct, UserStructHandler);
    MESSAGE_HANDLER(WM_TEXTMESSAGE, TMyTextMessage, UserTextHandler);
END_MESSAGE_MAP(TComponent)

*/

void __fastcall TForm4::UserIntHandler(TMyIntMessage &msg){
    MessageDlg("Int message received: " + IntToStr(*msg.lparam),
            mtInformation, TMsgDlgButtons() << mbOK, 0);

    *msg.lparam = *msg.lparam + 50;
}
void __fastcall TForm4::UserStructHandler(TMyStruct &msg){
    MessageDlg("Struct received: " + IntToStr(msg.lparam->m_a) +
```

```
                " " + IntToStr(msg.lparam->m_b),
            mtInformation, TMsgDlgButtons() << mbOK, 0);
}
void __fastcall TForm4::UserGetStructHandler(TMyStruct &msg){
    MessageDlg("Struct received: " + IntToStr(msg.lparam->m_a) +
            "" + IntToStr(msg.lparam->m_b),
            mtInformation, TMsgDlgButtons() << mbOK, 0);
    msg.lparam->m_a = msg.lparam->m_a + 50;
    msg.lparam->m_b = msg.lparam->m_b + 50;
}
void __fastcall TForm4::UserTextHandler(TMyTextMessage &msg){
    MessageDlg("Text received: " + msg.lparam,
            mtInformation, TMsgDlgButtons() << mbOK, 0);
}
void __fastcall TForm4::Button1Click(TObject *Sender)
{
    int wparam = 0;
    int lparam = StrToIntDef(IntEdit->Text,0);
    int lbase = lparam;

    SendGetIntMessage(Handle,WM_INTMESSAGE,wparam,lparam);

    if (lbase != lparam)
    {
        MessageDlg("Value modified by handler function: " +
                    IntToStr(lparam),
                    mtInformation, TMsgDlgButtons() << mbOK, 0);
        IntEdit->Text = IntToStr(lparam);
    }

}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button2Click(TObject *Sender){
    int wparam = 0;
    String lparam = TextEdit->Text;

    SendTextMessage(Handle,WM_TEXTMESSAGE,wparam,lparam);
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button3Click(TObject *Sender)
{
    int wparam = 0;
    MStruct lparam;

    lparam.m_a = StrToIntDef(StructEdit1->Text,0);
    lparam.m_b = StrToIntDef(StructEdit2->Text,0);

    SendStructMessage(Handle, WM_STRUCTMESSAGE, wparam, &lparam);
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button4Click(TObject *Sender)
{
    int wparam = 0;
    MStruct lparam, lbase;

    lparam.m_a = StrToIntDef(GetStructEdit1->Text,0);
    lparam.m_b = StrToIntDef(GetStructEdit2->Text,0);

    lbase = lparam;

    SendStructMessage(Handle, WM_GETSTRUCTMESSAGE, wparam, &lparam);

    if (lbase.m_a != lparam.m_a) {
```

```
        MessageDlg("Struct modified by handler function: " +
                   IntToStr(lparam.m_a) + " " + IntToStr(lparam.m_b),
                   mtInformation, TMsgDlgButtons() << mbOK, 0);
        GetStructEdit1->Text = IntToStr(lparam.m_a);
        GetStructEdit2->Text = IntToStr(lparam.m_b);
    }
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This is an example demonstrating the usage of the SendGetIntMessage,
SendStructMessage, SendGetStructMessage and SendTextMessage message
functions.
This example uses four buttons and six text edit boxes as follows:
- Button1 //SendGetIntMessage
- Button2 //SendStructMessage
- Button3 //SendGetStructMessage
- Button4 //SendTextMessage

- IntEdit //for int inputs
- StructEdit1 & StructEdit2 //for struct inputs (ints)
- GetStructEdit1 & GetStructEdit2 //for struct inputs (ints)
- TextEdit //for text input
}


const
  WM_INTMSG = WM_USER + 100;
  WM_STRUCTMSG = WM_USER + 101;
  WM_GETSTRUCTMSG = WM_USER + 102;
  WM_TEXTMSG = WM_USER + 103;


type
  TMyIntMessage = packed record
    Msg: Cardinal;
    WParam: ^Integer;
    LParam: ^Integer;
    Result: LRESULT;
  end;

  TMyTextMessage = packed record
    Msg: Cardinal;
    WParam: Integer;
    LParam: String;
    Result: LRESULT;
  end;

  MStruct = packed record
    m_a : Integer;
    m_b : Integer;
  end;

  TMyStruct = packed record
    Msg : Cardinal;
    WParam : Integer;
    LParam : ^MStruct;
    Result : LRESULT;
  end;

  TForm4 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
```

```
    IntEdit: TEdit;
    StructEdit1: TEdit;
    GetStructEdit1: TEdit;
    TextEdit: TEdit;
    StructEdit2: TEdit;
    GetStructEdit2: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure UserIntHandler(var msg: TMyIntMessage); message WM_INTMSG;

    procedure UserStructHandler(var msg: TMyStruct); message WM_STRUCTMSG;

    procedure UserGetStructHandler(var msg: TMyStruct); message WM_GETSTRUCTMSG;

    procedure UserTextHandler(var msg: TMyTextMessage); message WM_TEXTMSG;

  end;

var
  Form4: TForm4;

implementation

{$R *.dfm}

{ TForm4 }

procedure TForm4.Button1Click(Sender: TObject);
var
  wparam : Integer;
  lparam, lbase : Integer;
begin
  //SendGetIntMessage
  wparam := 0;

  lparam := StrToIntDef(IntEdit.Text,0);
  lbase := lparam;

  SendGetIntMessage(Handle, WM_INTMSG, wparam, lparam);

  if lbase <> lparam then
  begin
  MessageDlg('Value modified by handler function: ' + IntToStr(lparam),
             mtInformation, [mbOK], 0);
  IntEdit.Text := IntToStr(lparam);
  end;
end;

procedure TForm4.Button2Click(Sender: TObject);
var
  wparam : Integer;
  lparam : MStruct;
begin
  //SendStructMessage
  wparam := 0;

  lparam.m_a := StrToIntDef(StructEdit1.Text,0);
  lparam.m_b := StrToIntDef(StructEdit2.Text,0);

  SendStructMessage(Handle, WM_STRUCTMSG, wparam, lparam);
```

```
end;

procedure TForm4.Button3Click(Sender: TObject);
var
  wparam : Integer;
  lparam, lbase : MStruct;
begin
  //SendGetStructMessage
  wparam := 0;

  lparam.m_a := StrToIntDef(GetStructEdit1.Text,0);
  lparam.m_b := StrToIntDef(GetStructEdit2.Text,0);

  lbase := lparam;

  SendGetStructMessage(Handle, WM_GETSTRUCTMSG, wparam, lparam);

  if lbase.m_a <> lparam.m_a then
  begin
  MessageDlg('Struct modified by handler function: ' +
              IntToStr(lparam.m_a) + ' ' + IntToStr(lparam.m_b),
              mtInformation, [mbOK], 0);
  GetStructEdit1.Text := IntToStr(lparam.m_a);
  GetStructEdit2.Text := IntToStr(lparam.m_b);
  end;

end;

procedure TForm4.Button4Click(Sender: TObject);
var
  wparam : Integer;
  lparam : UnicodeString;
begin
  //SendTextMessage
  wparam := 0;

  lparam := TextEdit.Text;

  SendTextMessage(Handle, WM_TEXTMSG, wparam, lparam);

end;

procedure TForm4.UserGetStructHandler(var msg: TMyStruct);
begin
  MessageDlg('Struct msg received: ' + IntToStr(msg.LParam^.m_a) +
            ' '  + IntToStr(msg.LParam^.m_b), mtInformation, [mbOK], 0);
  msg.LParam^.m_a := msg.LParam^.m_a + 50;
  msg.LParam^.m_b := msg.LParam^.m_b + 50;
end;

procedure TForm4.UserIntHandler(var msg: TMyIntMessage);
begin
  MessageDlg('Int msg received: ' + IntToStr(msg.LParam^),
              mtInformation, [mbOK], 0);

  msg.LParam^ := msg.LParam^ + 50;
end;

procedure TForm4.UserStructHandler(var msg: TMyStruct);
begin
  MessageDlg('Struct msg received: ' + IntToStr(msg.LParam^.m_a) +
            ' '  + IntToStr(msg.LParam^.m_b), mtInformation, [mbOK], 0);
end;

procedure TForm4.UserTextHandler(var msg: TMyTextMessage);
begin
```

```
    MessageDlg('Text msg received: ' + msg.LParam,
               mtInformation, [mbOK], 0);
  end;
```

See Also

SendMessage (MSDN)

LPARAM (MSDN)

WPARAM (MSDN)

Handling messages

35.1.2 Messages.SendGetStructMessage Function

Send Windows message, sending and receiving data structure.

Pascal

```
function SendGetStructMessage(Handle: HWND; Msg: UINT; WParam: WPARAM; var LParam; Unused:
Boolean = False): LRESULT;
```

C++

```
LRESULT SendGetStructMessage(HWND Handle, UINT Msg, WPARAM WParam,  LParam, Boolean Unused =
False);
```

File

Messages

Description

SendGetStructMessage calls the Windows **SendMessage** function and returns an **LRESULT**.

Handle is an **HWND** window handle.

Msg is the Windows message ID. This constant is exactly the same as the Windows message ID described in the Microsoft Windows SDK.

WParam is a **WPARAM** for the message, depending on **Msg**. The value is sent.

LParam is a data structure **LPARAM** for the message, depending on **Msg**. A value may be sent and returned.

Unused is not used and is optional.

Various records have been defined in the **Messages (see page 1475)** unit to represent Windows messages. For instance, **TWMCopy (see page 1572)** represents the message **WM_COPY (see page 1686)**, **TWMKeyDown (see page 1579)** represents **WM_KEYDOWN (see page 1722)**, and so on.

See the Microsoft Windows SDK documentation for information on the Windows messages and their associated **WPARAM** and **LPARAM**.

C++ Examples:

```
/*
This is an example demonstrating the usage of the SendGetIntMessage,
SendStructMessage, SendGetStructMessage and SendTextMessage message
functions.
This example uses four buttons and six text edit boxes as follows:
- Button1 //SendGetIntMessage
- Button2 //SendStructMessage
- Button3 //SendGetStructMessage
```

```
- Button4 //SendTextMessage

- IntEdit //for int inputs
- StructEdit1 & StructEdit2 //for struct inputs (ints)
- GetStructEdit1 & GetStructEdit2 //for struct inputs (ints)
- TextEdit //for text input

Note: This example generates unresolved external errors with the
default NO_STRICT conditional define set.  The work around is
to use the STRICT conditional define.  This problem is addressed
in the RAID bug #267447.

This example uses three custom message structures defined in the header:
struct TMyIntMessage {
    Cardinal Msg;
    int* wparam;
    int* lparam;
    LRESULT Result;
};

struct TMyTextMessage {
    Cardinal Msg;
    int wparam;
    String lparam;
    LRESULT Result;
};

struct MStruct {
    int m_a;
    int m_b;
};

struct TMyStruct {
    Cardinal Msg;
    int wparam;
    MStruct* lparam;
    LRESULT Result;
};

Furthermore it uses 4 message handling functions described in the
header along with a message map:

MESSAGE void __fastcall UserIntHandler(TMyIntMessage &msg);
MESSAGE void __fastcall UserStructHandler(TMyStruct &msg);
MESSAGE void __fastcall UserGetStructHandler(TMyStruct &msg);
MESSAGE void __fastcall UserTextHandler(TMyTextMessage &msg);

BEGIN_MESSAGE_MAP
    MESSAGE_HANDLER(WM_INTMESSAGE, TMyIntMessage, UserIntHandler);
    MESSAGE_HANDLER(WM_GETSTRUCTMESSAGE, TMyStruct, UserGetStructHandler);
    MESSAGE_HANDLER(WM_STRUCTMESSAGE, TMyStruct, UserStructHandler);
    MESSAGE_HANDLER(WM_TEXTMESSAGE, TMyTextMessage, UserTextHandler);
END_MESSAGE_MAP(TComponent)

*/

void __fastcall TForm4::UserIntHandler(TMyIntMessage &msg){
    MessageDlg("Int message received: " + IntToStr(*msg.lparam),
            mtInformation, TMsgDlgButtons() << mbOK, 0);

    *msg.lparam = *msg.lparam + 50;
}
void __fastcall TForm4::UserStructHandler(TMyStruct &msg){
    MessageDlg("Struct received: " + IntToStr(msg.lparam->m_a) +
                " " + IntToStr(msg.lparam->m_b),
            mtInformation, TMsgDlgButtons() << mbOK, 0);
```

```
}
void __fastcall TForm4::UserGetStructHandler(TMyStruct &msg){
    MessageDlg("Struct received: " + IntToStr(msg.lparam->m_a) +
            "" + IntToStr(msg.lparam->m_b),
            mtInformation, TMsgDlgButtons() << mbOK, 0);
    msg.lparam->m_a = msg.lparam->m_a + 50;
    msg.lparam->m_b = msg.lparam->m_b + 50;
}
void __fastcall TForm4::UserTextHandler(TMyTextMessage &msg){
    MessageDlg("Text received: " + msg.lparam,
            mtInformation, TMsgDlgButtons() << mbOK, 0);
}
void __fastcall TForm4::Button1Click(TObject *Sender)
{
    int wparam = 0;
    int lparam = StrToIntDef(IntEdit->Text,0);
    int lbase = lparam;

    SendGetIntMessage(Handle,WM_INTMESSAGE,wparam,lparam);

    if (lbase != lparam)
    {
        MessageDlg("Value modified by handler function: " +
                    IntToStr(lparam),
                    mtInformation, TMsgDlgButtons() << mbOK, 0);
        IntEdit->Text = IntToStr(lparam);
    }

}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button2Click(TObject *Sender){
    int wparam = 0;
    String lparam = TextEdit->Text;

    SendTextMessage(Handle,WM_TEXTMESSAGE,wparam,lparam);
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button3Click(TObject *Sender)
{
    int wparam = 0;
    MStruct lparam;

    lparam.m_a = StrToIntDef(StructEdit1->Text,0);
    lparam.m_b = StrToIntDef(StructEdit2->Text,0);

    SendStructMessage(Handle, WM_STRUCTMESSAGE, wparam, &lparam);
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button4Click(TObject *Sender)
{
    int wparam = 0;
    MStruct lparam, lbase;

    lparam.m_a = StrToIntDef(GetStructEdit1->Text,0);
    lparam.m_b = StrToIntDef(GetStructEdit2->Text,0);

    lbase = lparam;

    SendStructMessage(Handle, WM_GETSTRUCTMESSAGE, wparam, &lparam);

    if (lbase.m_a != lparam.m_a) {
        MessageDlg("Struct modified by handler function: " +
                    IntToStr(lparam.m_a) + " " + IntToStr(lparam.m_b),
```

```
                    mtInformation, TMsgDlgButtons() << mbOK, 0);
        GetStructEdit1->Text = IntToStr(lparam.m_a);
        GetStructEdit2->Text = IntToStr(lparam.m_b);
    }
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This is an example demonstrating the usage of the SendGetIntMessage,
SendStructMessage, SendGetStructMessage and SendTextMessage message
functions.
This example uses four buttons and six text edit boxes as follows:
- Button1 //SendGetIntMessage
- Button2 //SendStructMessage
- Button3 //SendGetStructMessage
- Button4 //SendTextMessage

- IntEdit //for int inputs
- StructEdit1 & StructEdit2 //for struct inputs (ints)
- GetStructEdit1 & GetStructEdit2 //for struct inputs (ints)
- TextEdit //for text input
}


const
  WM_INTMSG = WM_USER + 100;
  WM_STRUCTMSG = WM_USER + 101;
  WM_GETSTRUCTMSG = WM_USER + 102;
  WM_TEXTMSG = WM_USER + 103;


type
  TMyIntMessage = packed record
    Msg: Cardinal;
    WParam: ^Integer;
    LParam: ^Integer;
    Result: LRESULT;
  end;

  TMyTextMessage = packed record
    Msg: Cardinal;
    WParam: Integer;
    LParam: String;
    Result: LRESULT;
  end;

  MStruct = packed record
    m_a : Integer;
    m_b : Integer;
  end;

  TMyStruct = packed record
    Msg : Cardinal;
    WParam : Integer;
    LParam : ^MStruct;
    Result : LRESULT;
  end;

  TForm4 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    IntEdit: TEdit;
    StructEdit1: TEdit;
```

```
    GetStructEdit1: TEdit;
    TextEdit: TEdit;
    StructEdit2: TEdit;
    GetStructEdit2: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure UserIntHandler(var msg: TMyIntMessage); message WM_INTMSG;

    procedure UserStructHandler(var msg: TMyStruct); message WM_STRUCTMSG;

    procedure UserGetStructHandler(var msg: TMyStruct); message WM_GETSTRUCTMSG;

    procedure UserTextHandler(var msg: TMyTextMessage); message WM_TEXTMSG;

  end;

var
  Form4: TForm4;

implementation

{$R *.dfm}

{ TForm4 }

procedure TForm4.Button1Click(Sender: TObject);
var
  wparam : Integer;
  lparam, lbase : Integer;
begin
  //SendGetIntMessage
  wparam := 0;

  lparam := StrToIntDef(IntEdit.Text,0);
  lbase := lparam;

  SendGetIntMessage(Handle, WM_INTMSG, wparam, lparam);

  if lbase <> lparam then
  begin
  MessageDlg('Value modified by handler function: ' + IntToStr(lparam),
             mtInformation, [mbOK], 0);
  IntEdit.Text := IntToStr(lparam);
  end;
end;

procedure TForm4.Button2Click(Sender: TObject);
var
  wparam : Integer;
  lparam : MStruct;
begin
  //SendStructMessage
  wparam := 0;

  lparam.m_a := StrToIntDef(StructEdit1.Text,0);
  lparam.m_b := StrToIntDef(StructEdit2.Text,0);

  SendStructMessage(Handle, WM_STRUCTMSG, wparam, lparam);
end;
```

```
procedure TForm4.Button3Click(Sender: TObject);
var
  wparam : Integer;
  lparam, lbase : MStruct;
begin
  //SendGetStructMessage
  wparam := 0;

  lparam.m_a := StrToIntDef(GetStructEdit1.Text,0);
  lparam.m_b := StrToIntDef(GetStructEdit2.Text,0);

  lbase := lparam;

  SendGetStructMessage(Handle, WM_GETSTRUCTMSG, wparam, lparam);

  if lbase.m_a <> lparam.m_a then
  begin
  MessageDlg('Struct modified by handler function: ' +
             IntToStr(lparam.m_a) + ' ' + IntToStr(lparam.m_b),
             mtInformation, [mbOK], 0);
  GetStructEdit1.Text := IntToStr(lparam.m_a);
  GetStructEdit2.Text := IntToStr(lparam.m_b);
  end;

end;

procedure TForm4.Button4Click(Sender: TObject);
var
  wparam : Integer;
  lparam : UnicodeString;
begin
  //SendTextMessage
  wparam := 0;

  lparam := TextEdit.Text;

  SendTextMessage(Handle, WM_TEXTMSG, wparam, lparam);

end;

procedure TForm4.UserGetStructHandler(var msg: TMyStruct);
begin
  MessageDlg('Struct msg received: ' + IntToStr(msg.LParam^.m_a) +
           ' '  + IntToStr(msg.LParam^.m_b), mtInformation, [mbOK], 0);
  msg.LParam^.m_a := msg.LParam^.m_a + 50;
  msg.LParam^.m_b := msg.LParam^.m_b + 50;
end;

procedure TForm4.UserIntHandler(var msg: TMyIntMessage);
begin
  MessageDlg('Int msg received: ' + IntToStr(msg.LParam^),
             mtInformation, [mbOK], 0);

  msg.LParam^ := msg.LParam^ + 50;
end;

procedure TForm4.UserStructHandler(var msg: TMyStruct);
begin
  MessageDlg('Struct msg received: ' + IntToStr(msg.LParam^.m_a) +
           ' '  + IntToStr(msg.LParam^.m_b), mtInformation, [mbOK], 0);
end;

procedure TForm4.UserTextHandler(var msg: TMyTextMessage);
begin
  MessageDlg('Text msg received: ' + msg.LParam,
             mtInformation, [mbOK], 0);
```

```
end;
```

See Also

SendMessage (MSDN)

LPARAM (MSDN)

WPARAM (MSDN)

Handling messages

35.1.3 Messages.SendStructMessage Function

Send Windows message, sending data structure.

Pascal

```
function SendStructMessage(Handle: HWND; Msg: UINT; WParam: WPARAM; const LParam): LRESULT;
```

C++

```
LRESULT SendStructMessage(HWND Handle, UINT Msg, WPARAM WParam, const  LParam);
```

File

Messages

Description

SendStructMessage calls the Windows **SendMessage** function and returns an **LRESULT**.

Handle is an **HWND** window handle.

Msg is the Windows message ID. This constant is exactly the same as the Windows message ID described in the Microsoft Windows SDK.

WParam is a **WPARAM** for the message, depending on **Msg**. The value is sent.

LParam is a data structure **LPARAM** for the message, depending on **Msg**. The value is sent.

Various records have been defined in the **Messages (see page 1475)** unit to represent Windows messages. For instance, **TWMCopy (see page 1572)** represents the message **WM_COPY (see page 1686)**, **TWMKeyDown (see page 1579)** represents **WM_KEYDOWN (see page 1722)**, and so on.

See the Microsoft Windows SDK documentation for information on the Windows messages and their associated **WPARAM** and **LPARAM**.

C++ Examples:

```
/*
This is an example demonstrating the usage of the SendGetIntMessage,
SendStructMessage, SendGetStructMessage and SendTextMessage message
functions.
This example uses four buttons and six text edit boxes as follows:
- Button1 //SendGetIntMessage
- Button2 //SendStructMessage
- Button3 //SendGetStructMessage
- Button4 //SendTextMessage

- IntEdit //for int inputs
- StructEdit1 & StructEdit2 //for struct inputs (ints)
- GetStructEdit1 & GetStructEdit2 //for struct inputs (ints)
- TextEdit //for text input
```

```
Note: This example generates unresolved external errors with the
default NO_STRICT conditional define set.  The work around is
to use the STRICT conditional define.  This problem is addressed
in the RAID bug #267447.

This example uses three custom message structures defined in the header:
struct TMyIntMessage {
    Cardinal Msg;
    int* wparam;
    int* lparam;
    LRESULT Result;
};

struct TMyTextMessage {
    Cardinal Msg;
    int wparam;
    String lparam;
    LRESULT Result;
};

struct MStruct {
    int m_a;
    int m_b;
};

struct TMyStruct {
    Cardinal Msg;
    int wparam;
    MStruct* lparam;
    LRESULT Result;
};

Furthermore it uses 4 message handling functions described in the
header along with a message map:

MESSAGE void __fastcall UserIntHandler(TMyIntMessage &msg);
MESSAGE void __fastcall UserStructHandler(TMyStruct &msg);
MESSAGE void __fastcall UserGetStructHandler(TMyStruct &msg);
MESSAGE void __fastcall UserTextHandler(TMyTextMessage &msg);

BEGIN_MESSAGE_MAP
    MESSAGE_HANDLER(WM_INTMESSAGE, TMyIntMessage, UserIntHandler);
    MESSAGE_HANDLER(WM_GETSTRUCTMESSAGE, TMyStruct, UserGetStructHandler);
    MESSAGE_HANDLER(WM_STRUCTMESSAGE, TMyStruct, UserStructHandler);
    MESSAGE_HANDLER(WM_TEXTMESSAGE, TMyTextMessage, UserTextHandler);
END_MESSAGE_MAP(TComponent)

*/

void __fastcall TForm4::UserIntHandler(TMyIntMessage &msg){
    MessageDlg("Int message received: " + IntToStr(*msg.lparam),
            mtInformation, TMsgDlgButtons() << mbOK, 0);

    *msg.lparam = *msg.lparam + 50;
}
void __fastcall TForm4::UserStructHandler(TMyStruct &msg){
    MessageDlg("Struct received: " + IntToStr(msg.lparam->m_a) +
                " " + IntToStr(msg.lparam->m_b),
            mtInformation, TMsgDlgButtons() << mbOK, 0);
}
void __fastcall TForm4::UserGetStructHandler(TMyStruct &msg){
    MessageDlg("Struct received: " + IntToStr(msg.lparam->m_a) +
            "" + IntToStr(msg.lparam->m_b),
            mtInformation, TMsgDlgButtons() << mbOK, 0);
    msg.lparam->m_a = msg.lparam->m_a + 50;
```

```
    msg.lparam->m_b = msg.lparam->m_b + 50;
}
void __fastcall TForm4::UserTextHandler(TMyTextMessage &msg){
    MessageDlg("Text received: " + msg.lparam,
            mtInformation, TMsgDlgButtons() << mbOK, 0);
}
void __fastcall TForm4::Button1Click(TObject *Sender)
{
    int wparam = 0;
    int lparam = StrToIntDef(IntEdit->Text,0);
    int lbase = lparam;

    SendGetIntMessage(Handle,WM_INTMESSAGE,wparam,lparam);

    if (lbase != lparam)
    {
        MessageDlg("Value modified by handler function: " +
                    IntToStr(lparam),
                    mtInformation, TMsgDlgButtons() << mbOK, 0);
        IntEdit->Text = IntToStr(lparam);
    }

}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button2Click(TObject *Sender){
    int wparam = 0;
    String lparam = TextEdit->Text;

    SendTextMessage(Handle,WM_TEXTMESSAGE,wparam,lparam);
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button3Click(TObject *Sender)
{
    int wparam = 0;
    MStruct lparam;

    lparam.m_a = StrToIntDef(StructEdit1->Text,0);
    lparam.m_b = StrToIntDef(StructEdit2->Text,0);

    SendStructMessage(Handle, WM_STRUCTMESSAGE, wparam, &lparam);
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button4Click(TObject *Sender)
{
    int wparam = 0;
    MStruct lparam, lbase;

    lparam.m_a = StrToIntDef(GetStructEdit1->Text,0);
    lparam.m_b = StrToIntDef(GetStructEdit2->Text,0);

    lbase = lparam;

    SendStructMessage(Handle, WM_GETSTRUCTMESSAGE, wparam, &lparam);

    if (lbase.m_a != lparam.m_a) {
        MessageDlg("Struct modified by handler function: " +
                    IntToStr(lparam.m_a) + " " + IntToStr(lparam.m_b),
                    mtInformation, TMsgDlgButtons() << mbOK, 0);
        GetStructEdit1->Text = IntToStr(lparam.m_a);
        GetStructEdit2->Text = IntToStr(lparam.m_b);
    }
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This is an example demonstrating the usage of the SendGetIntMessage,
SendStructMessage, SendGetStructMessage and SendTextMessage message
functions.
This example uses four buttons and six text edit boxes as follows:
- Button1 //SendGetIntMessage
- Button2 //SendStructMessage
- Button3 //SendGetStructMessage
- Button4 //SendTextMessage

- IntEdit //for int inputs
- StructEdit1 & StructEdit2 //for struct inputs (ints)
- GetStructEdit1 & GetStructEdit2 //for struct inputs (ints)
- TextEdit //for text input
}


const
  WM_INTMSG = WM_USER + 100;
  WM_STRUCTMSG = WM_USER + 101;
  WM_GETSTRUCTMSG = WM_USER + 102;
  WM_TEXTMSG = WM_USER + 103;


type
  TMyIntMessage = packed record
    Msg: Cardinal;
    WParam: ^Integer;
    LParam: ^Integer;
    Result: LRESULT;
  end;

  TMyTextMessage = packed record
    Msg: Cardinal;
    WParam: Integer;
    LParam: String;
    Result: LRESULT;
  end;

  MStruct = packed record
    m_a : Integer;
    m_b : Integer;
  end;

  TMyStruct = packed record
    Msg : Cardinal;
    WParam : Integer;
    LParam : ^MStruct;
    Result : LRESULT;
  end;

  TForm4 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    IntEdit: TEdit;
    StructEdit1: TEdit;
    GetStructEdit1: TEdit;
    TextEdit: TEdit;
    StructEdit2: TEdit;
    GetStructEdit2: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
```

```
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure UserIntHandler(var msg: TMyIntMessage); message WM_INTMSG;

    procedure UserStructHandler(var msg: TMyStruct); message WM_STRUCTMSG;

    procedure UserGetStructHandler(var msg: TMyStruct); message WM_GETSTRUCTMSG;

    procedure UserTextHandler(var msg: TMyTextMessage); message WM_TEXTMSG;

  end;

var
  Form4: TForm4;

implementation

{$R *.dfm}

{ TForm4 }

procedure TForm4.Button1Click(Sender: TObject);
var
  wparam : Integer;
  lparam, lbase : Integer;
begin
  //SendGetIntMessage
  wparam := 0;

  lparam := StrToIntDef(IntEdit.Text,0);
  lbase := lparam;

  SendGetIntMessage(Handle, WM_INTMSG, wparam, lparam);

  if lbase <> lparam then
  begin
  MessageDlg('Value modified by handler function: ' + IntToStr(lparam),
             mtInformation, [mbOK], 0);
  IntEdit.Text := IntToStr(lparam);
  end;
end;

procedure TForm4.Button2Click(Sender: TObject);
var
  wparam : Integer;
  lparam : MStruct;
begin
  //SendStructMessage
  wparam := 0;

  lparam.m_a := StrToIntDef(StructEdit1.Text,0);
  lparam.m_b := StrToIntDef(StructEdit2.Text,0);

  SendStructMessage(Handle, WM_STRUCTMSG, wparam, lparam);
end;

procedure TForm4.Button3Click(Sender: TObject);
var
  wparam : Integer;
  lparam, lbase : MStruct;
begin
  //SendGetStructMessage
```

```
    wparam := 0;

    lparam.m_a := StrToIntDef(GetStructEdit1.Text,0);
    lparam.m_b := StrToIntDef(GetStructEdit2.Text,0);

    lbase := lparam;

    SendGetStructMessage(Handle, WM_GETSTRUCTMSG, wparam, lparam);

    if lbase.m_a <> lparam.m_a then
    begin
    MessageDlg('Struct modified by handler function: ' +
                IntToStr(lparam.m_a) + ' ' + IntToStr(lparam.m_b),
                mtInformation, [mbOK], 0);
    GetStructEdit1.Text := IntToStr(lparam.m_a);
    GetStructEdit2.Text := IntToStr(lparam.m_b);
    end;

  end;

  procedure TForm4.Button4Click(Sender: TObject);
  var
    wparam : Integer;
    lparam : UnicodeString;
  begin
    //SendTextMessage
    wparam := 0;

    lparam := TextEdit.Text;

    SendTextMessage(Handle, WM_TEXTMSG, wparam, lparam);

  end;

  procedure TForm4.UserGetStructHandler(var msg: TMyStruct);
  begin
    MessageDlg('Struct msg received: ' + IntToStr(msg.LParam^.m_a) +
              ' '  + IntToStr(msg.LParam^.m_b), mtInformation, [mbOK], 0);
    msg.LParam^.m_a := msg.LParam^.m_a + 50;
    msg.LParam^.m_b := msg.LParam^.m_b + 50;
  end;

  procedure TForm4.UserIntHandler(var msg: TMyIntMessage);
  begin
    MessageDlg('Int msg received: ' + IntToStr(msg.LParam^),
                mtInformation, [mbOK], 0);

    msg.LParam^ := msg.LParam^ + 50;
  end;

  procedure TForm4.UserStructHandler(var msg: TMyStruct);
  begin
    MessageDlg('Struct msg received: ' + IntToStr(msg.LParam^.m_a) +
              ' '  + IntToStr(msg.LParam^.m_b), mtInformation, [mbOK], 0);
  end;

  procedure TForm4.UserTextHandler(var msg: TMyTextMessage);
  begin
    MessageDlg('Text msg received: ' + msg.LParam,
                mtInformation, [mbOK], 0);
  end;
```

See Also

SendMessage (MSDN)

LPARAM (MSDN)

WPARAM (MSDN)

Handling messages

35.1.4 Messages.SendTextMessage Function

Send Windows message, sending text.

Pascal

```
function SendTextMessage(Handle: HWND; Msg: UINT; WParam: WPARAM; LParam: UnicodeString):
LRESULT; overload;
```

C++

```
LRESULT SendTextMessage(HWND Handle, UINT Msg, WPARAM WParam, UnicodeString LParam);
```

File

Messages

Description

SendTextMessage calls the Windows **SendMessage** function and returns an **LRESULT**.

Handle is an **HWND** window handle.

Msg is the Windows message ID. This constant is exactly the same as the Windows message ID described in the Microsoft Windows SDK.

WParam is a **WPARAM** for the message, depending on **Msg**. The value is sent.

LParam is a **string LPARAM** for the message, depending on **Msg**. The value is sent.

Various records have been defined in the **Messages (see page 1475)** unit to represent Windows messages. For instance, **TWMCopy (see page 1572)** represents the message **WM_COPY (see page 1686)**, **TWMKeyDown (see page 1579)** represents **WM_KEYDOWN (see page 1722)**, and so on.

See the Microsoft Windows SDK documentation for information on the Windows messages and their associated **WPARAM** and **LPARAM**.

C++ Examples:

```
/*
This is an example demonstrating the usage of the SendGetIntMessage,
SendStructMessage, SendGetStructMessage and SendTextMessage message
functions.
This example uses four buttons and six text edit boxes as follows:
- Button1 //SendGetIntMessage
- Button2 //SendStructMessage
- Button3 //SendGetStructMessage
- Button4 //SendTextMessage

- IntEdit //for int inputs
- StructEdit1 & StructEdit2 //for struct inputs (ints)
- GetStructEdit1 & GetStructEdit2 //for struct inputs (ints)
- TextEdit //for text input

Note: This example generates unresolved external errors with the
default NO_STRICT conditional define set.  The work around is
to use the STRICT conditional define.  This problem is addressed
in the RAID bug #267447.
```

```
  This example uses three custom message structures defined in the header:
  struct TMyIntMessage {
      Cardinal Msg;
      int* wparam;
      int* lparam;
      LRESULT Result;
  };

  struct TMyTextMessage {
      Cardinal Msg;
      int wparam;
      String lparam;
      LRESULT Result;
  };

  struct MStruct {
      int m_a;
      int m_b;
  };

  struct TMyStruct {
      Cardinal Msg;
      int wparam;
      MStruct* lparam;
      LRESULT Result;
  };

  Furthermore it uses 4 message handling functions described in the
  header along with a message map:

  MESSAGE void __fastcall UserIntHandler(TMyIntMessage &msg);
  MESSAGE void __fastcall UserStructHandler(TMyStruct &msg);
  MESSAGE void __fastcall UserGetStructHandler(TMyStruct &msg);
  MESSAGE void __fastcall UserTextHandler(TMyTextMessage &msg);

  BEGIN_MESSAGE_MAP
      MESSAGE_HANDLER(WM_INTMESSAGE, TMyIntMessage, UserIntHandler);
      MESSAGE_HANDLER(WM_GETSTRUCTMESSAGE, TMyStruct, UserGetStructHandler);
      MESSAGE_HANDLER(WM_STRUCTMESSAGE, TMyStruct, UserStructHandler);
      MESSAGE_HANDLER(WM_TEXTMESSAGE, TMyTextMessage, UserTextHandler);
  END_MESSAGE_MAP(TComponent)

  */

  void __fastcall TForm4::UserIntHandler(TMyIntMessage &msg){
      MessageDlg("Int message received: " + IntToStr(*msg.lparam),
              mtInformation, TMsgDlgButtons() << mbOK, 0);

      *msg.lparam = *msg.lparam + 50;
  }
  void __fastcall TForm4::UserStructHandler(TMyStruct &msg){
      MessageDlg("Struct received: " + IntToStr(msg.lparam->m_a) +
                  " " + IntToStr(msg.lparam->m_b),
              mtInformation, TMsgDlgButtons() << mbOK, 0);
  }
  void __fastcall TForm4::UserGetStructHandler(TMyStruct &msg){
      MessageDlg("Struct received: " + IntToStr(msg.lparam->m_a) +
              "" + IntToStr(msg.lparam->m_b),
              mtInformation, TMsgDlgButtons() << mbOK, 0);
      msg.lparam->m_a = msg.lparam->m_a + 50;
      msg.lparam->m_b = msg.lparam->m_b + 50;
  }
  void __fastcall TForm4::UserTextHandler(TMyTextMessage &msg){
      MessageDlg("Text received: " + msg.lparam,
              mtInformation, TMsgDlgButtons() << mbOK, 0);
  }
```

```
void __fastcall TForm4::Button1Click(TObject *Sender)
{
    int wparam = 0;
    int lparam = StrToIntDef(IntEdit->Text,0);
    int lbase = lparam;

    SendGetIntMessage(Handle,WM_INTMESSAGE,wparam,lparam);

    if (lbase != lparam)
    {
        MessageDlg("Value modified by handler function: " +
                   IntToStr(lparam),
                   mtInformation, TMsgDlgButtons() << mbOK, 0);
        IntEdit->Text = IntToStr(lparam);
    }

}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button2Click(TObject *Sender){
    int wparam = 0;
    String lparam = TextEdit->Text;

    SendTextMessage(Handle,WM_TEXTMESSAGE,wparam,lparam);
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button3Click(TObject *Sender)
{
    int wparam = 0;
    MStruct lparam;

    lparam.m_a = StrToIntDef(StructEdit1->Text,0);
    lparam.m_b = StrToIntDef(StructEdit2->Text,0);

    SendStructMessage(Handle, WM_STRUCTMESSAGE, wparam, &lparam);
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button4Click(TObject *Sender)
{
    int wparam = 0;
    MStruct lparam, lbase;

    lparam.m_a = StrToIntDef(GetStructEdit1->Text,0);
    lparam.m_b = StrToIntDef(GetStructEdit2->Text,0);

    lbase = lparam;

    SendStructMessage(Handle, WM_GETSTRUCTMESSAGE, wparam, &lparam);

    if (lbase.m_a != lparam.m_a) {
        MessageDlg("Struct modified by handler function: " +
                   IntToStr(lparam.m_a) + " " + IntToStr(lparam.m_b),
                   mtInformation, TMsgDlgButtons() << mbOK, 0);
        GetStructEdit1->Text = IntToStr(lparam.m_a);
        GetStructEdit2->Text = IntToStr(lparam.m_b);
    }
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This is an example demonstrating the usage of the SendGetIntMessage,
SendStructMessage, SendGetStructMessage and SendTextMessage message
functions.
```

```
  This example uses four buttons and six text edit boxes as follows:
  - Button1 //SendGetIntMessage
  - Button2 //SendStructMessage
  - Button3 //SendGetStructMessage
  - Button4 //SendTextMessage

  - IntEdit //for int inputs
  - StructEdit1 & StructEdit2 //for struct inputs (ints)
  - GetStructEdit1 & GetStructEdit2 //for struct inputs (ints)
  - TextEdit //for text input
  }


  const
    WM_INTMSG = WM_USER + 100;
    WM_STRUCTMSG = WM_USER + 101;
    WM_GETSTRUCTMSG = WM_USER + 102;
    WM_TEXTMSG = WM_USER + 103;


  type
    TMyIntMessage = packed record
      Msg: Cardinal;
      WParam: ^Integer;
      LParam: ^Integer;
      Result: LRESULT;
    end;

    TMyTextMessage = packed record
      Msg: Cardinal;
      WParam: Integer;
      LParam: String;
      Result: LRESULT;
    end;

    MStruct = packed record
      m_a : Integer;
      m_b : Integer;
    end;

    TMyStruct = packed record
      Msg : Cardinal;
      WParam : Integer;
      LParam : ^MStruct;
      Result : LRESULT;
    end;

    TForm4 = class(TForm)
      Button1: TButton;
      Button2: TButton;
      Button3: TButton;
      Button4: TButton;
      IntEdit: TEdit;
      StructEdit1: TEdit;
      GetStructEdit1: TEdit;
      TextEdit: TEdit;
      StructEdit2: TEdit;
      GetStructEdit2: TEdit;
      procedure Button1Click(Sender: TObject);
      procedure Button4Click(Sender: TObject);
      procedure Button2Click(Sender: TObject);
      procedure Button3Click(Sender: TObject);
    private
      { Private declarations }
    public
      { Public declarations }
```

```
    procedure UserIntHandler(var msg: TMyIntMessage); message WM_INTMSG;

    procedure UserStructHandler(var msg: TMyStruct); message WM_STRUCTMSG;

    procedure UserGetStructHandler(var msg: TMyStruct); message WM_GETSTRUCTMSG;

    procedure UserTextHandler(var msg: TMyTextMessage); message WM_TEXTMSG;

  end;

var
  Form4: TForm4;

implementation

{$R *.dfm}

{ TForm4 }

procedure TForm4.Button1Click(Sender: TObject);
var
  wparam : Integer;
  lparam, lbase : Integer;
begin
  //SendGetIntMessage
  wparam := 0;

  lparam := StrToIntDef(IntEdit.Text,0);
  lbase := lparam;

  SendGetIntMessage(Handle, WM_INTMSG, wparam, lparam);

  if lbase <> lparam then
  begin
  MessageDlg('Value modified by handler function: ' + IntToStr(lparam),
             mtInformation, [mbOK], 0);
  IntEdit.Text := IntToStr(lparam);
  end;
end;

procedure TForm4.Button2Click(Sender: TObject);
var
  wparam : Integer;
  lparam : MStruct;
begin
  //SendStructMessage
  wparam := 0;

  lparam.m_a := StrToIntDef(StructEdit1.Text,0);
  lparam.m_b := StrToIntDef(StructEdit2.Text,0);

  SendStructMessage(Handle, WM_STRUCTMSG, wparam, lparam);
end;

procedure TForm4.Button3Click(Sender: TObject);
var
  wparam : Integer;
  lparam, lbase : MStruct;
begin
  //SendGetStructMessage
  wparam := 0;

  lparam.m_a := StrToIntDef(GetStructEdit1.Text,0);
  lparam.m_b := StrToIntDef(GetStructEdit2.Text,0);

  lbase := lparam;
```

```
    SendGetStructMessage(Handle, WM_GETSTRUCTMSG, wparam, lparam);

    if lbase.m_a <> lparam.m_a then
    begin
    MessageDlg('Struct modified by handler function: ' +
                IntToStr(lparam.m_a) + ' ' + IntToStr(lparam.m_b),
                mtInformation, [mbOK], 0);
    GetStructEdit1.Text := IntToStr(lparam.m_a);
    GetStructEdit2.Text := IntToStr(lparam.m_b);
    end;

  end;

  procedure TForm4.Button4Click(Sender: TObject);
  var
    wparam : Integer;
    lparam : UnicodeString;
  begin
    //SendTextMessage
    wparam := 0;

    lparam := TextEdit.Text;

    SendTextMessage(Handle, WM_TEXTMSG, wparam, lparam);

  end;

  procedure TForm4.UserGetStructHandler(var msg: TMyStruct);
  begin
    MessageDlg('Struct msg received: ' + IntToStr(msg.LParam^.m_a) +
              ' '  + IntToStr(msg.LParam^.m_b), mtInformation, [mbOK], 0);
    msg.LParam^.m_a := msg.LParam^.m_a + 50;
    msg.LParam^.m_b := msg.LParam^.m_b + 50;
  end;

  procedure TForm4.UserIntHandler(var msg: TMyIntMessage);
  begin
    MessageDlg('Int msg received: ' + IntToStr(msg.LParam^),
                mtInformation, [mbOK], 0);

    msg.LParam^ := msg.LParam^ + 50;
  end;

  procedure TForm4.UserStructHandler(var msg: TMyStruct);
  begin
    MessageDlg('Struct msg received: ' + IntToStr(msg.LParam^.m_a) +
              ' '  + IntToStr(msg.LParam^.m_b), mtInformation, [mbOK], 0);
  end;

  procedure TForm4.UserTextHandler(var msg: TMyTextMessage);
  begin
    MessageDlg('Text msg received: ' + msg.LParam,
                mtInformation, [mbOK], 0);
  end;
```

See Also

SendMessage (MSDN)

LPARAM (MSDN)

WPARAM (MSDN)

Handling messages

35.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Records

	Name	Description
	TMessage (see page 1513)	TMessage represents a Windows message.
	TMSHMouseWheel (see page 1514)	This is record Messages.TMSHMouseWheel.
	TWMActivate (see page 1514)	This is record Messages.TWMActivate.
	TWMActivateApp (see page 1515)	This is record Messages.TWMActivateApp.
	TWMAskCBFormatName (see page 1515)	This is record Messages.TWMAskCBFormatName.
	TWMChangeCBChain (see page 1516)	This is record Messages.TWMChangeCBChain.
	TWMCharToItem (see page 1516)	This is record Messages.TWMCharToItem.
	TWMChooseFont_GetLogFont (see page 1517)	This is record Messages.TWMChooseFont_GetLogFont.
	TWMCommand (see page 1517)	This is record Messages.TWMCommand.
	TWMCompacting (see page 1518)	This is record Messages.TWMCompacting.
	TWMCompareItem (see page 1518)	This is record Messages.TWMCompareItem.
	TWMContextMenu (see page 1519)	This is record Messages.TWMContextMenu.
	TWMCopyData (see page 1519)	This is record Messages.TWMCopyData.
	TWMCreate (see page 1520)	This is record Messages.TWMCreate.
	TWMCtlColor (see page 1520)	This is record Messages.TWMCtlColor.
	TWMDDE_Ack (see page 1521)	This is record Messages.TWMDDE_Ack.
	TWMDDE_Advise (see page 1521)	This is record Messages.TWMDDE_Advise.
	TWMDDE_Data (see page 1522)	This is record Messages.TWMDDE_Data.
	TWMDDE_Execute (see page 1522)	This is record Messages.TWMDDE_Execute.
	TWMDDE_Initiate (see page 1523)	This is record Messages.TWMDDE_Initiate.
	TWMDDE_Poke (see page 1523)	This is record Messages.TWMDDE_Poke.
	TWMDDE_Request (see page 1524)	This is record Messages.TWMDDE_Request.
	TWMDDE_Terminate (see page 1524)	This is record Messages.TWMDDE_Terminate.
	TWMDDE_Unadvise (see page 1525)	This is record Messages.TWMDDE_Unadvise.
	TWMDeleteItem (see page 1525)	This is record Messages.TWMDeleteItem.
	TWMDevModeChange (see page 1526)	This is record Messages.TWMDevModeChange.
	TWMDisplayChange (see page 1526)	This is record Messages.TWMDisplayChange.
	TWMDrawItem (see page 1527)	This is record Messages.TWMDrawItem.
	TWMDropFiles (see page 1527)	This is record Messages.TWMDropFiles.
	TWMEnable (see page 1528)	This is record Messages.TWMEnable.
	TWMEndSession (see page 1528)	This is record Messages.TWMEndSession.
	TWMEnterIdle (see page 1529)	This is record Messages.TWMEnterIdle.
	TWMEnterMenuLoop (see page 1529)	This is record Messages.TWMEnterMenuLoop.
	TWMEraseBkgnd (see page 1530)	This is record Messages.TWMEraseBkgnd.
	TWMGetIcon (see page 1530)	This is record Messages.TWMGetIcon.
	TWMGetMinMaxInfo (see page 1531)	This is record Messages.TWMGetMinMaxInfo.
	TWMGetText (see page 1531)	This is record Messages.TWMGetText.
	TWMHelp (see page 1532)	This is record Messages.TWMHelp.
	TWMHotKey (see page 1532)	This is record Messages.TWMHotKey.
	TWMHScrollClipboard (see page 1533)	This is record Messages.TWMHScrollClipboard.
	TWMInitDialog (see page 1533)	This is record Messages.TWMInitDialog.
	TWMInitMenu (see page 1534)	This is record Messages.TWMInitMenu.
	TWMInitMenuPopup (see page 1534)	This is record Messages.TWMInitMenuPopup.
	TWMKey (see page 1535)	TWMKey represents a Windows Keystroke message.

	TWMKillFocus (see page 1536)	This is record Messages.TWMKillFocus.
	TWMMDIActivate (see page 1536)	This is record Messages.TWMMDIActivate.
	TWMMDICascade (see page 1537)	This is record Messages.TWMMDICascade.
	TWMMDICreate (see page 1537)	This is record Messages.TWMMDICreate.
	TWMMDIDestroy (see page 1538)	This is record Messages.TWMMDIDestroy.
	TWMMDIMaximize (see page 1538)	This is record Messages.TWMMDIMaximize.
	TWMMDINext (see page 1539)	This is record Messages.TWMMDINext.
	TWMMDIRestore (see page 1539)	This is record Messages.TWMMDIRestore.
	TWMMDISetMenu (see page 1540)	This is record Messages.TWMMDISetMenu.
	TWMMDITile (see page 1540)	This is record Messages.TWMMDITile.
	TWMMeasureItem (see page 1541)	This is record Messages.TWMMeasureItem.
	TWMMenuChar (see page 1541)	TWMMenuChar represents a Windows Keystroke message directed at a menu.
	TWMMenuSelect (see page 1542)	This is record Messages.TWMMenuSelect.
	TWMMouse (see page 1543)	TWMMouse represents a Windows mouse message.
	TWMMouseActivate (see page 1544)	This is record Messages.TWMMouseActivate.
	TWMMouseWheel (see page 1544)	This is record Messages.TWMMouseWheel.
	TWMMove (see page 1545)	This is record Messages.TWMMove.
	TWMMoving (see page 1545)	This is record Messages.TWMMoving.
	TWMNCActivate (see page 1546)	This is record Messages.TWMNCActivate.
	TWMNCCalcSize (see page 1546)	This is record Messages.TWMNCCalcSize.
	TWMNCCreate (see page 1547)	This is record Messages.TWMNCCreate.
	TWMNCHitMessage (see page 1547)	This is record Messages.TWMNCHitMessage.
	TWMNCHitTest (see page 1548)	This is record Messages.TWMNCHitTest.
	TWMNCPaint (see page 1548)	This is record Messages.TWMNCPaint.
	TWMNextDlgCtl (see page 1549)	This is record Messages.TWMNextDlgCtl.
	TWMNoParams (see page 1549)	This is record Messages.TWMNoParams.
	TWMNotify (see page 1550)	TWMNotify represents a Windows notification message.
	TWMNotifyFormat (see page 1550)	This is record Messages.TWMNotifyFormat.
	TWMPaint (see page 1551)	TWMPaint represents a Windows Paint message.
	TWMPaintClipboard (see page 1551)	This is record Messages.TWMPaintClipboard.
	TWMPaletteChanged (see page 1552)	This is record Messages.TWMPaletteChanged.
	TWMPaletteIsChanging (see page 1552)	This is record Messages.TWMPaletteIsChanging.
	TWMParentNotify (see page 1553)	This is record Messages.TWMParentNotify.
	TWMPower (see page 1554)	This is record Messages.TWMPower.
	TWMPrint (see page 1554)	This is record Messages.TWMPrint.
	TWMQueryEndSession (see page 1555)	This is record Messages.TWMQueryEndSession.
	TWMQuit (see page 1555)	This is record Messages.TWMQuit.
	TWMRenderFormat (see page 1556)	This is record Messages.TWMRenderFormat.
	TWMScroll (see page 1556)	This is record Messages.TWMScroll.
	TWMSetCursor (see page 1557)	This is record Messages.TWMSetCursor.
	TWMSetFocus (see page 1557)	This is record Messages.TWMSetFocus.
	TWMSetFont (see page 1558)	This is record Messages.TWMSetFont.
	TWMSetHotKey (see page 1558)	This is record Messages.TWMSetHotKey.
	TWMSetIcon (see page 1559)	This is record Messages.TWMSetIcon.
	TWMSetRedraw (see page 1559)	This is record Messages.TWMSetRedraw.
	TWMSetText (see page 1560)	This is record Messages.TWMSetText.
	TWMSettingChange (see page 1560)	This is record Messages.TWMSettingChange.
	TWMShowWindow (see page 1561)	This is record Messages.TWMShowWindow.
	TWMSize (see page 1561)	This is record Messages.TWMSize.
	TWMSizeClipboard (see page 1562)	This is record Messages.TWMSizeClipboard.
	TWMSpoolerStatus (see page 1562)	This is record Messages.TWMSpoolerStatus.
	TWMStyleChange (see page 1563)	This is record Messages.TWMStyleChange.
	TWMSysCommand (see page 1563)	This is record Messages.TWMSysCommand.
	TWMSysDeadChar (see page 1564)	This is record Messages.TWMSysDeadChar.

	TWMSystemError (see page 1564)	This is record Messages.TWMSystemError.
	TWMTimer (see page 1565)	This is record Messages.TWMTimer.
	TWMUIState (see page 1565)	This is record Messages.TWMUIState.
	TWMVScrollClipboard (see page 1566)	This is record Messages.TWMVScrollClipboard.
	TWMWindowPosMsg (see page 1566)	This is record Messages.TWMWindowPosMsg.
	TWMWinIniChange (see page 1567)	This is record Messages.TWMWinIniChange.

35.2.1 Messages.TMessage Record

TMessage represents a Windows message.

Pascal

```
TMessage = packed record
  Msg: Cardinal;
  case Integer of
    0: (WParam: WPARAM;
        LParam: LPARAM;
        Result: LRESULT;);
    1: (WParamLo: Word;
        WParamHi: Word;
        LParamLo: Word;
        LParamHi: Word;
        ResultLo: Word;
        ResultHi: Word;);
end;
```

C++

```
struct TMessage {
  unsigned Msg;
  WPARAM WParam;
  LPARAM LParam;
  LRESULT Result;
  Word WParamLo;
  Word WParamHi;
  Word LParamLo;
  Word LParamHi;
  Word ResultLo;
  Word ResultHi;
};
```

File

Messages

Description

The TMessage type represents a Windows message in WndProc and other methods.

The TMessage contains the following fields:

Field	Meaning
Msg	Represents the Windows Message code.
WParam	Represents the WParam of the message. To access the low and high words of this field, use the WParamLo and WParamHi fields instead.
LParam	Represents the LParam of the message. To access the low and high words of this field, use the LParamLo and LParamHi fields instead.
Result	Holds the return value. To access the low and high words of this field, use the ResultLo and ResultHi fields instead.

See Also

TWMKey (see page 1535)

TWMMouse (see page 1543)

TWMPaint (see page 1551)

TWMMenuChar (see page 1541)

TWMNotify (see page 1550)

TWMLButtonDown (see page 1580)

TMsg

35.2.2 Messages.TMSHMouseWheel Record

Pascal

```
TMSHMouseWheel = packed record
  Msg: Cardinal;
  WheelDelta: Integer;
  case Integer of
    0: (XPos: Smallint;
        YPos: Smallint;);
    1: (Pos: TSmallPoint;
        Result: Longint;);
end;
```

C++

```
struct TMSHMouseWheel {
  unsigned Msg;
  int WheelDelta;
  Smallint XPos;
  Smallint YPos;
  TSmallPoint Pos;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TMSHMouseWheel.

35.2.3 Messages.TWMActivate Record

Pascal

```
TWMActivate = packed record
  Msg: Cardinal;
  Active: Word;
  Minimized: WordBool;
  ActiveWindow: HWND;
  Result: Longint;
end;
```

C++

```
struct TWMActivate {
  unsigned Msg;
  Word Active;
  WordBool Minimized;
  HWND ActiveWindow;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMActivate.

35.2.4 Messages.TWMActivateApp Record

Pascal

```
TWMActivateApp = packed record
  Msg: Cardinal;
  Active: BOOL;
  ThreadId: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMActivateApp {
  unsigned Msg;
  BOOL Active;
  Longint ThreadId;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMActivateApp.

35.2.5 Messages.TWMAskCBFormatName Record

Pascal

```
TWMAskCBFormatName = packed record
  Msg: Cardinal;
  NameLen: Word;
  Unused: Word;
  FormatName: PChar;
  Result: Longint;
end;
```

C++

```
struct TWMAskCBFormatName {
```

```
    unsigned Msg;
    Word NameLen;
    Word Unused;
    const char * FormatName;
    Longint Result;
  };
```

File

Messages

Description

This is record Messages.TWMAskCBFormatName.

35.2.6 Messages.TWMChangeCBChain Record

Pascal

```
  TWMChangeCBChain = packed record
    Msg: Cardinal;
    Remove: HWND;
    Next: HWND;
    Result: Longint;
  end;
```

C++

```
  struct TWMChangeCBChain {
    unsigned Msg;
    HWND Remove;
    HWND Next;
    Longint Result;
  };
```

File

Messages

Description

This is record Messages.TWMChangeCBChain.

35.2.7 Messages.TWMCharToItem Record

Pascal

```
  TWMCharToItem = packed record
    Msg: Cardinal;
    Key: Word;
    CaretPos: Word;
    ListBox: HWND;
    Result: Longint;
  end;
```

C++

```
  struct TWMCharToItem {
    unsigned Msg;
    Word Key;
    Word CaretPos;
    HWND ListBox;
```

```
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMCharToItem.

35.2.8 Messages.TWMChooseFont_GetLogFont Record

Pascal

```
TWMChooseFont_GetLogFont = packed record
  Msg: Cardinal;
  Unused: Longint;
  LogFont: PLogFont;
  Result: Longint;
end;
```

C++

```
struct TWMChooseFont_GetLogFont {
  unsigned Msg;
  Longint Unused;
  PLogFont LogFont;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMChooseFont_GetLogFont.

35.2.9 Messages.TWMCommand Record

Pascal

```
TWMCommand = packed record
  Msg: Cardinal;
  ItemID: Word;
  NotifyCode: Word;
  Ctl: HWND;
  Result: Longint;
end;
```

C++

```
struct TWMCommand {
  unsigned Msg;
  Word ItemID;
  Word NotifyCode;
  HWND Ctl;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMCommand.

35.2.10 Messages.TWMCompacting Record

Pascal

```
TWMCompacting = packed record
  Msg: Cardinal;
  CompactRatio: Longint;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMCompacting {
  unsigned Msg;
  Longint CompactRatio;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMCompacting.

35.2.11 Messages.TWMCompareItem Record

Pascal

```
TWMCompareItem = packed record
  Msg: Cardinal;
  Ctl: HWnd;
  CompareItemStruct: PCompareItemStruct;
  Result: Longint;
end;
```

C++

```
struct TWMCompareItem {
  unsigned Msg;
  HWnd Ctl;
  PCompareItemStruct CompareItemStruct;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMCompareItem.

35.2.12 Messages.TWMContextMenu Record

Pascal

```
TWMContextMenu = packed record
  Msg: Cardinal;
  hWnd: HWND;
  case Integer of
    0: (XPos: Smallint;
        YPos: Smallint;);
    1: (Pos: TSmallPoint;
        Result: Longint;);
end;
```

C++

```
struct TWMContextMenu {
  unsigned Msg;
  HWND hWnd;
  Smallint XPos;
  Smallint YPos;
  TSmallPoint Pos;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMContextMenu.

35.2.13 Messages.TWMCopyData Record

Pascal

```
TWMCopyData = packed record
  Msg: Cardinal;
  From: HWND;
  CopyDataStruct: PCopyDataStruct;
  Result: Longint;
end;
```

C++

```
struct TWMCopyData {
  unsigned Msg;
  HWND From;
  PCopyDataStruct CopyDataStruct;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMCopyData.

35.2.14 Messages.TWMCreate Record

Pascal

```
TWMCreate = packed record
  Msg: Cardinal;
  Unused: Integer;
  CreateStruct: PCreateStruct;
  Result: Longint;
end;
```

C++

```
struct TWMCreate {
  unsigned Msg;
  int Unused;
  PCreateStruct CreateStruct;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMCreate.

35.2.15 Messages.TWMCtlColor Record

Pascal

```
TWMCtlColor = packed record
  Msg: Cardinal;
  ChildDC: HDC;
  ChildWnd: HWND;
  Result: Longint;
end;
```

C++

```
struct TWMCtlColor {
  unsigned Msg;
  HDC ChildDC;
  HWND ChildWnd;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMCtlColor.

35.2.16 Messages.TWMDDE_Ack Record

Pascal

```
TWMDDE_Ack = packed record
  Msg: Cardinal;
  PostingApp: HWND;
  case Word of
    WM_DDE_EXECUTE: (PackedVal: Longint;);
    WM_DDE_INITIATE: (App: Word;
                      Topic: Word;
                      Result: Longint;);
end;
```

C++

```
struct TWMDDE_Ack {
  unsigned Msg;
  HWND PostingApp;
  Word App;
  Word Topic;
  Longint Result;
  Longint PackedVal;
};
```

File

Messages

Description

This is record Messages.TWMDDE_Ack.

35.2.17 Messages.TWMDDE_Advise Record

Pascal

```
TWMDDE_Advise = packed record
  Msg: Cardinal;
  PostingApp: HWND;
  PackedVal: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMDDE_Advise {
  unsigned Msg;
  HWND PostingApp;
  Longint PackedVal;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDDE_Advise.

35.2.18 Messages.TWMDDE_Data Record

Pascal

```
TWMDDE_Data = packed record
  Msg: Cardinal;
  PostingApp: HWND;
  PackedVal: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMDDE_Data {
  unsigned Msg;
  HWND PostingApp;
  Longint PackedVal;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDDE_Data.

35.2.19 Messages.TWMDDE_Execute Record

Pascal

```
TWMDDE_Execute = packed record
  Msg: Cardinal;
  PostingApp: HWND;
  Commands: THandle;
  Result: Longint;
end;
```

C++

```
struct TWMDDE_Execute {
  unsigned Msg;
  HWND PostingApp;
  THandle Commands;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDDE_Execute.

35.2.20 Messages.TWMDDE_Initiate Record

Pascal

```
TWMDDE_Initiate = packed record
  Msg: Cardinal;
  PostingApp: HWND;
  App: Word;
  Topic: Word;
  Result: Longint;
end;
```

C++

```
struct TWMDDE_Initiate {
  unsigned Msg;
  HWND PostingApp;
  Word App;
  Word Topic;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDDE_Initiate.

35.2.21 Messages.TWMDDE_Poke Record

Pascal

```
TWMDDE_Poke = packed record
  Msg: Cardinal;
  PostingApp: HWND;
  PackedVal: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMDDE_Poke {
  unsigned Msg;
  HWND PostingApp;
  Longint PackedVal;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDDE_Poke.

35.2.22 Messages.TWMDDE_Request Record

Pascal

```
TWMDDE_Request = packed record
  Msg: Cardinal;
  PostingApp: HWND;
  Format: Word;
  Item: Word;
  Result: Longint;
end;
```

C++

```
struct TWMDDE_Request {
  unsigned Msg;
  HWND PostingApp;
  Word Format;
  Word Item;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDDE_Request.

35.2.23 Messages.TWMDDE_Terminate Record

Pascal

```
TWMDDE_Terminate = packed record
  Msg: Cardinal;
  PostingApp: HWND;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMDDE_Terminate {
  unsigned Msg;
  HWND PostingApp;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDDE_Terminate.

35.2.24 Messages.TWMDDE_Unadvise Record

Pascal

```
TWMDDE_Unadvise = packed record
  Msg: Cardinal;
  PostingApp: HWND;
  Format: Word;
  Item: Word;
  Result: Longint;
end;
```

C++

```
struct TWMDDE_Unadvise {
  unsigned Msg;
  HWND PostingApp;
  Word Format;
  Word Item;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDDE_Unadvise.

35.2.25 Messages.TWMDeleteItem Record

Pascal

```
TWMDeleteItem = packed record
  Msg: Cardinal;
  Ctl: HWND;
  DeleteItemStruct: PDeleteItemStruct;
  Result: Longint;
end;
```

C++

```
struct TWMDeleteItem {
  unsigned Msg;
  HWND Ctl;
  PDeleteItemStruct DeleteItemStruct;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDeleteItem.

35.2.26 Messages.TWMDevModeChange Record

Pascal

```
TWMDevModeChange = packed record
  Msg: Cardinal;
  Unused: Integer;
  Device: PChar;
  Result: Longint;
end;
```

C++

```
struct TWMDevModeChange {
  unsigned Msg;
  int Unused;
  const char * Device;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDevModeChange.

35.2.27 Messages.TWMDisplayChange Record

Pascal

```
TWMDisplayChange = packed record
  Msg: Cardinal;
  BitsPerPixel: Integer;
  Width: Word;
  Height: Word;
  Result: Longint;
end;
```

C++

```
struct TWMDisplayChange {
  unsigned Msg;
  int BitsPerPixel;
  Word Width;
  Word Height;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDisplayChange.

35.2.28 Messages.TWMDrawItem Record

Pascal

```
TWMDrawItem = packed record
  Msg: Cardinal;
  Ctl: HWND;
  DrawItemStruct: PDrawItemStruct;
  Result: Longint;
end;
```

C++

```
struct TWMDrawItem {
  unsigned Msg;
  HWND Ctl;
  PDrawItemStruct DrawItemStruct;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDrawItem.

35.2.29 Messages.TWMDropFiles Record

Pascal

```
TWMDropFiles = packed record
  Msg: Cardinal;
  Drop: THANDLE;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMDropFiles {
  unsigned Msg;
  THANDLE Drop;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMDropFiles.

35.2.30 Messages.TWMEnable Record

Pascal

```
TWMEnable = packed record
  Msg: Cardinal;
  Enabled: LongBool;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMEnable {
  unsigned Msg;
  LongBool Enabled;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMEnable.

35.2.31 Messages.TWMEndSession Record

Pascal

```
TWMEndSession = packed record
  Msg: Cardinal;
  EndSession: LongBool;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMEndSession {
  unsigned Msg;
  LongBool EndSession;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMEndSession.

35.2.32 Messages.TWMEnterIdle Record

Pascal

```
TWMEnterIdle = packed record
  Msg: Cardinal;
  Source: Longint;
  IdleWnd: HWND;
  Result: Longint;
end;
```

C++

```
struct TWMEnterIdle {
  unsigned Msg;
  Longint Source;
  HWND IdleWnd;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMEnterIdle.

35.2.33 Messages.TWMEnterMenuLoop Record

Pascal

```
TWMEnterMenuLoop = packed record
  Msg: Cardinal;
  IsTrackPopupMenu: LongBool;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMEnterMenuLoop {
  unsigned Msg;
  LongBool IsTrackPopupMenu;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMEnterMenuLoop.

35.2.34 Messages.TWMEraseBkgnd Record

Pascal

```
TWMEraseBkgnd = packed record
  Msg: Cardinal;
  DC: HDC;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMEraseBkgnd {
  unsigned Msg;
  HDC DC;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMEraseBkgnd.

35.2.35 Messages.TWMGetIcon Record

Pascal

```
TWMGetIcon = packed record
  Msg: Cardinal;
  BigIcon: Longbool;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMGetIcon {
  unsigned Msg;
  Longbool BigIcon;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMGetIcon.

35.2.36 Messages.TWMGetMinMaxInfo Record

Pascal

```
TWMGetMinMaxInfo = packed record
  Msg: Cardinal;
  Unused: Integer;
  MinMaxInfo: PMinMaxInfo;
  Result: Longint;
end;
```

C++

```
struct TWMGetMinMaxInfo {
  unsigned Msg;
  int Unused;
  PMinMaxInfo MinMaxInfo;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMGetMinMaxInfo.

35.2.37 Messages.TWMGetText Record

Pascal

```
TWMGetText = packed record
  Msg: Cardinal;
  TextMax: Integer;
  Text: PChar;
  Result: Longint;
end;
```

C++

```
struct TWMGetText {
  unsigned Msg;
  int TextMax;
  const char * Text;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMGetText.

35.2.38 Messages.TWMHelp Record

Pascal

```
TWMHelp = packed record
  Msg: Cardinal;
  Unused: Integer;
  HelpInfo: PHelpInfo;
  Result: Longint;
end;
```

C++

```
struct TWMHelp {
  unsigned Msg;
  int Unused;
  PHelpInfo HelpInfo;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMHelp.

35.2.39 Messages.TWMHotKey Record

Pascal

```
TWMHotKey = packed record
  Msg: Cardinal;
  HotKey: Longint;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMHotKey {
  unsigned Msg;
  Longint HotKey;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMHotKey.

35.2.40 Messages.TWMHScrollClipboard Record

Pascal

```
TWMHScrollClipboard = packed record
  Msg: Cardinal;
  Viewer: HWND;
  ScrollCode: Word;
  Pos: Word;
  Result: Longint;
end;
```

C++

```
struct TWMHScrollClipboard {
  unsigned Msg;
  HWND Viewer;
  Word ScrollCode;
  Word Pos;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMHScrollClipboard.

35.2.41 Messages.TWMInitDialog Record

Pascal

```
TWMInitDialog = packed record
  Msg: Cardinal;
  Focus: HWND;
  InitParam: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMInitDialog {
  unsigned Msg;
  HWND Focus;
  Longint InitParam;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMInitDialog.

35.2.42 Messages.TWMInitMenu Record

Pascal

```
TWMInitMenu = packed record
  Msg: Cardinal;
  Menu: HMENU;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMInitMenu {
  unsigned Msg;
  HMENU Menu;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMInitMenu.

35.2.43 Messages.TWMInitMenuPopup Record

Pascal

```
TWMInitMenuPopup = packed record
  Msg: Cardinal;
  MenuPopup: HMENU;
  Pos: Smallint;
  SystemMenu: WordBool;
  Result: Longint;
end;
```

C++

```
struct TWMInitMenuPopup {
  unsigned Msg;
  HMENU MenuPopup;
  Smallint Pos;
  WordBool SystemMenu;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMInitMenuPopup.

35.2.44 Messages.TWMKey Record

TWMKey represents a Windows Keystroke message.

Pascal

```
TWMKey = packed record
  Msg: Cardinal;
  CharCode: Word;
  Unused: Word;
  KeyData: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMKey {
  unsigned Msg;
  Word CharCode;
  Word Unused;
  Longint KeyData;
  Longint Result;
};
```

File

Messages

Description

TWMKey is the data structure used in some Windows keyboard messages.

TWMKey contains the following fields.

Field	Meaning
Msg	Represents the Windows message code.
CharCode	Specifies the virtual key code for the key pressed by the user. This is either an ANSI character or a virtual key constant, such as VK_CANCEL.
Unused	Unused.
KeyData	Indicates the repeat count, scan code, extended-key flag, context code and so on. See the Windows help system for a keystroke message such as WM_KEYDOWN (see page 1722) for details on the possible values.
Result	Holds the value returned by the application processing the message.

See Also

TWMMouse (see page 1543)

TWMPaint (see page 1551)

TWMMenuChar (see page 1541)

TWMNotify (see page 1550)

TWMLButtonDown (see page 1580)

TMsg

35.2.45 Messages.TWMKillFocus Record

Pascal

```
TWMKillFocus = packed record
  Msg: Cardinal;
  FocusedWnd: HWND;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMKillFocus {
  unsigned Msg;
  HWND FocusedWnd;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMKillFocus.

35.2.46 Messages.TWMMDIActivate Record

Pascal

```
TWMMDIActivate = packed record
  Msg: Cardinal;
  case Integer of
    0: (ChildWnd: HWND;);
    1: (DeactiveWnd: HWND;
        ActiveWnd: HWND;
        Result: Longint;);
end;
```

C++

```
struct TWMMDIActivate {
  unsigned Msg;
  HWND ChildWnd;
  HWND DeactiveWnd;
  HWND ActiveWnd;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMDIActivate.

35.2.47 Messages.TWMMDICascade Record

Pascal

```
TWMMDICascade = packed record
  Msg: Cardinal;
  Cascade: Longint;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMMDICascade {
  unsigned Msg;
  Longint Cascade;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMDICascade.

35.2.48 Messages.TWMMDICreate Record

Pascal

```
TWMMDICreate = packed record
  Msg: Cardinal;
  Unused: Integer;
  MDICreateStruct: PMDICreateStruct;
  Result: Longint;
end;
```

C++

```
struct TWMMDICreate {
  unsigned Msg;
  int Unused;
  PMDICreateStruct MDICreateStruct;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMDICreate.

35.2.49 Messages.TWMMDIDestroy Record

Pascal

```
TWMMDIDestroy = packed record
  Msg: Cardinal;
  Child: HWND;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMMDIDestroy {
  unsigned Msg;
  HWND Child;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMDIDestroy.

35.2.50 Messages.TWMMDIMaximize Record

Pascal

```
TWMMDIMaximize = packed record
  Msg: Cardinal;
  Maximize: HWND;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMMDIMaximize {
  unsigned Msg;
  HWND Maximize;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMDIMaximize.

35.2.51 Messages.TWMMDINext Record

Pascal

```
TWMMDINext = packed record
  Msg: Cardinal;
  Child: HWND;
  Next: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMMDINext {
  unsigned Msg;
  HWND Child;
  Longint Next;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMDINext.

35.2.52 Messages.TWMMDIRestore Record

Pascal

```
TWMMDIRestore = packed record
  Msg: Cardinal;
  IDChild: HWND;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMMDIRestore {
  unsigned Msg;
  HWND IDChild;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMDIRestore.

35.2.53 Messages.TWMMDISetMenu Record

Pascal

```
TWMMDISetMenu = packed record
  Msg: Cardinal;
  MenuFrame: HMENU;
  MenuWindow: HMENU;
  Result: Longint;
end;
```

C++

```
struct TWMMDISetMenu {
  unsigned Msg;
  HMENU MenuFrame;
  HMENU MenuWindow;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMDISetMenu.

35.2.54 Messages.TWMMDITile Record

Pascal

```
TWMMDITile = packed record
  Msg: Cardinal;
  Tile: Longint;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMMDITile {
  unsigned Msg;
  Longint Tile;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMDITile.

35.2.55 Messages.TWMMeasureItem Record

Pascal

```
TWMMeasureItem = packed record
  Msg: Cardinal;
  IDCtl: HWnd;
  MeasureItemStruct: PMeasureItemStruct;
  Result: Longint;
end;
```

C++

```
struct TWMMeasureItem {
  unsigned Msg;
  HWnd IDCtl;
  PMeasureItemStruct MeasureItemStruct;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMeasureItem.

35.2.56 Messages.TWMMenuChar Record

TWMMenuChar represents a Windows Keystroke message directed at a menu.

Pascal

```
TWMMenuChar = packed record
  Msg: Cardinal;
  User: Char;
  MenuFlag: Word;
  Menu: HMENU;
  Result: Longint;
end;
```

C++

```
struct TWMMenuChar {
  unsigned Msg;
  Char User;
  Word MenuFlag;
  HMENU Menu;
  Longint Result;
};
```

File

Messages

Description

TWMMenuChar is the data structure used for all Windows keyboard messages that are sent to menu items. The `Msg` field is the Windows message code. The `User` field specifies the virtual key code for the key pressed by the user. This is either a character or a virtual key constant, such as `VK_CANCEL`. The `Unused` field is ignored. The `MenuFlag` field indicates the type of menu to

which the message is directed. It should always be MF_POPUP, indicating a drop-down menu, submenu, or shortcut menu. (Earlier versions of Windows also supported MF_SYSMENU for the system menu, but the use of this flag is obsolete). The Menu field is the handle of the menu to which the message is directed. The Result field holds the value returned by the application processing the message. The following table lists the possible values and their meaning.

Value	Meaning
MNC_IGNORE	Tells Windows to discard the character the user pressed and create a short beep on the system speaker.
MNC_CLOSE	Tells Windows to close the active menu.
MNC_EXECUTE	Combined with an index in the low-order word, tells Windows to choose the item specified by that index. The owner window receives a WM_COMMAND (see page 1684) message.
MNC_SELECT	Combined with an index in the low-order word, tells Windows to select the item specified by that index.

See Also

TWMPaint (see page 1551)

TWMNotify (see page 1550)

TWMMouse (see page 1543)

TWMLButtonDown (see page 1580)

35.2.57 Messages.TWMMenuSelect Record

Pascal

```
TWMMenuSelect = packed record
  Msg: Cardinal;
  IDItem: Word;
  MenuFlag: Word;
  Menu: HMENU;
  Result: Longint;
end;
```

C++

```
struct TWMMenuSelect {
  unsigned Msg;
  Word IDItem;
  Word MenuFlag;
  HMENU Menu;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMenuSelect.

35.2.58 Messages.TWMMouse Record

TWMMouse represents a Windows mouse message.

Pascal

```
TWMMouse = packed record
  Msg: Cardinal;
  Keys: Longint;
  case Integer of
    0: (XPos: Smallint;
        YPos: Smallint;);
    1: (Pos: TSmallPoint;
        Result: Longint;);
end;
```

C++

```
struct TWMMouse {
  unsigned Msg;
  Longint Keys;
  Smallint XPos;
  Smallint YPos;
  TSmallPoint Pos;
  Longint Result;
};
```

File

Messages

Description

TWMMouse is the data structure used for all Windows mouse messages. The `Msg` field is the Windows message code. The `Keys` field indicates whether various keys are down. It is a combination of the values listed in the following table.

Value	Meaning
`MK_CONTROL`	The Control key is down.
`MK_LBUTTON`	The left mouse button is down.
`MK_MBUTTON`	The middle mouse button is down.
`MK_RBUTTON`	The right mouse button is down.
`MK_SHIFT`	The Shift key is down.

The `Pos` field (or the `XPos` and `YPos` fields) indicates the global coordinates of the mouse pointer. The `Result` field holds the value returned by the application processing the message.

See Also

TWMPaint (see page 1551)

TWMNotify (see page 1550)

TWMMenuChar (see page 1541)

TWMLButtonDown (see page 1580)

35.2.59 Messages.TWMMouseActivate Record

Pascal

```
TWMMouseActivate = packed record
  Msg: Cardinal;
  TopLevel: HWND;
  HitTestCode: Word;
  MouseMsg: Word;
  Result: Longint;
end;
```

C++

```
struct TWMMouseActivate {
  unsigned Msg;
  HWND TopLevel;
  Word HitTestCode;
  Word MouseMsg;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMouseActivate.

35.2.60 Messages.TWMMouseWheel Record

Pascal

```
TWMMouseWheel = packed record
  Msg: Cardinal;
  Keys: SmallInt;
  WheelDelta: SmallInt;
  case Integer of
    0: (XPos: Smallint;
        YPos: Smallint;);
    1: (Pos: TSmallPoint;
        Result: Longint;);
end;
```

C++

```
struct TWMMouseWheel {
  unsigned Msg;
  SmallInt Keys;
  SmallInt WheelDelta;
  Smallint XPos;
  Smallint YPos;
  TSmallPoint Pos;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMouseWheel.

35.2.61 Messages.TWMMove Record

Pascal

```
TWMMove = packed record
  Msg: Cardinal;
  Unused: Integer;
  case Integer of
    0: (XPos: Smallint;
        YPos: Smallint;);
    1: (Pos: TSmallPoint;
        Result: Longint;);
end;
```

C++

```
struct TWMMove {
  unsigned Msg;
  int Unused;
  Smallint XPos;
  Smallint YPos;
  TSmallPoint Pos;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMMove.

35.2.62 Messages.TWMMoving Record

Pascal

```
TWMMoving = packed record
  Msg: Cardinal;
  Edge: Integer;
  DragRect: PRect;
end;
```

C++

```
struct TWMMoving {
  unsigned Msg;
  int Edge;
  PRect DragRect;
};
```

File

Messages

Description

This is record Messages.TWMMoving.

35.2.63 Messages.TWMNCActivate Record

Pascal

```
TWMNCActivate = packed record
  Msg: Cardinal;
  Active: BOOL;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMNCActivate {
  unsigned Msg;
  BOOL Active;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMNCActivate.

35.2.64 Messages.TWMNCCalcSize Record

Pascal

```
TWMNCCalcSize = packed record
  Msg: Cardinal;
  CalcValidRects: BOOL;
  CalcSize_Params: PNCCalcSizeParams;
  Result: Longint;
end;
```

C++

```
struct TWMNCCalcSize {
  unsigned Msg;
  BOOL CalcValidRects;
  PNCCalcSizeParams CalcSize_Params;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMNCCalcSize.

35.2.65 Messages.TWMNCCreate Record

Pascal

```
TWMNCCreate = packed record
  Msg: Cardinal;
  Unused: Integer;
  CreateStruct: PCreateStruct;
  Result: Longint;
end;
```

C++

```
struct TWMNCCreate {
  unsigned Msg;
  int Unused;
  PCreateStruct CreateStruct;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMNCCreate.

35.2.66 Messages.TWMNCHitMessage Record

Pascal

```
TWMNCHitMessage = packed record
  Msg: Cardinal;
  HitTest: Longint;
  XCursor: Smallint;
  YCursor: Smallint;
  Result: Longint;
end;
```

C++

```
struct TWMNCHitMessage {
  unsigned Msg;
  Longint HitTest;
  Smallint XCursor;
  Smallint YCursor;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMNCHitMessage.

35.2.67 Messages.TWMNCHitTest Record

Pascal

```
TWMNCHitTest = packed record
  Msg: Cardinal;
  Unused: Longint;
  case Integer of
    0: (XPos: Smallint;
        YPos: Smallint;);
    1: (Pos: TSmallPoint;
        Result: Longint;);
end;
```

C++

```
struct TWMNCHitTest {
  unsigned Msg;
  Longint Unused;
  Smallint XPos;
  Smallint YPos;
  TSmallPoint Pos;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMNCHitTest.

35.2.68 Messages.TWMNCPaint Record

Pascal

```
TWMNCPaint = packed record
  Msg: Cardinal;
  RGN: HRGN;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMNCPaint {
  unsigned Msg;
  HRGN RGN;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMNCPaint.

35.2.69 Messages.TWMNextDlgCtl Record

Pascal

```
TWMNextDlgCtl = packed record
  Msg: Cardinal;
  CtlFocus: Longint;
  Handle: WordBool;
  Unused: Word;
  Result: Longint;
end;
```

C++

```
struct TWMNextDlgCtl {
  unsigned Msg;
  Longint CtlFocus;
  WordBool Handle;
  Word Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMNextDlgCtl.

35.2.70 Messages.TWMNoParams Record

Pascal

```
TWMNoParams = packed record
  Msg: Cardinal;
  Unused: array[0..3] of Word;
  Result: Longint;
end;
```

C++

```
struct TWMNoParams {
  unsigned Msg;
  array[0..3] of Word Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMNoParams.

35.2.71 Messages.TWMNotify Record

TWMNotify represents a Windows notification message.

Pascal

```
TWMNotify = packed record
  Msg: Cardinal;
  IDCtrl: Longint;
  NMHdr: PNMHdr;
  Result: Longint;
end;
```

C++

```
struct TWMNotify {
  unsigned Msg;
  Longint IDCtrl;
  PNMHdr NMHdr;
  Longint Result;
};
```

File

Messages

Description

TWMNotify is the data structure used for all Windows notification messages. The `Msg` field is the Windows message code. The `IDCtrl` field is the resource ID of the control that sent the notification. The `NMHdr` field is a pointer to a notification header. This data structure has three members: `hwndFrom`, which is the window handle of the control that sent the notification, `idFrom`, which is the same as the `IDCtrl` field of TWMNotify, and `code`, which is the notification code. The notification code's meaning depends on the type of control that sent the message. The `Result` field holds the value returned by the application processing the message.

See Also

TWMMouse (see page 1543)

TWMPaint (see page 1551)

TWMMenuChar (see page 1541)

TWMLButtonDown (see page 1580)

35.2.72 Messages.TWMNotifyFormat Record

Pascal

```
TWMNotifyFormat = packed record
  Msg: Cardinal;
  From: HWND;
  Command: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMNotifyFormat {
  unsigned Msg;
```

```
  HWND From;
  Longint Command;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMNotifyFormat.

35.2.73 Messages.TWMPaint Record

TWMPaint represents a Windows Paint message.

Pascal

```
TWMPaint = packed record
  Msg: Cardinal;
  DC: HDC;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMPaint {
  unsigned Msg;
  HDC DC;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

TWMPaint is the data structure used for all Windows Paint messages. The `Msg` field is the Windows message code. The `DC` field specifies handle to the device context that should be used for painting. This value should be assigned as the Handle property of the canvas of the object that needs painting. If `DC` is null, a valid device context must be obtained by calling the Windows API BeginPaint. The `Unused` field is not used. The `Result` field holds the value returned by the application processing the message.

See Also

TWMMouse (see page 1543)

TWMNotify (see page 1550)

TWMMenuChar (see page 1541)

TWMLButtonDown (see page 1580)

35.2.74 Messages.TWMPaintClipboard Record

Pascal

```
TWMPaintClipboard = packed record
```

```
    Msg: Cardinal;
    Viewer: HWND;
    PaintStruct: THandle;
    Result: Longint;
  end;
```

C++

```
  struct TWMPaintClipboard {
    unsigned Msg;
    HWND Viewer;
    THandle PaintStruct;
    Longint Result;
  };
```

File

Messages

Description

This is record Messages.TWMPaintClipboard.

35.2.75 Messages.TWMPaletteChanged Record

Pascal

```
  TWMPaletteChanged = packed record
    Msg: Cardinal;
    PalChg: HWND;
    Unused: Longint;
    Result: Longint;
  end;
```

C++

```
  struct TWMPaletteChanged {
    unsigned Msg;
    HWND PalChg;
    Longint Unused;
    Longint Result;
  };
```

File

Messages

Description

This is record Messages.TWMPaletteChanged.

35.2.76 Messages.TWMPaletteIsChanging Record

Pascal

```
  TWMPaletteIsChanging = packed record
    Msg: Cardinal;
    Realize: HWND;
    Unused: Longint;
    Result: Longint;
  end;
```

C++

```
struct TWMPaletteIsChanging {
  unsigned Msg;
  HWND Realize;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMPaletteIsChanging.

35.2.77 Messages.TWMParentNotify Record

Pascal

```
TWMParentNotify = packed record
  Msg: Cardinal;
  Event: Word;
  case Word of
    0: (Value1: Word;
        Value2: Longint;
        Result: Longint;);
    WM_DESTROY: (ChildID: Word;
                 ChildWnd: HWnd;);
    WM_RBUTTONDOWN: (Value: Word;
                     XPos: Smallint;
                     YPos: Smallint;);
end;
```

C++

```
struct TWMParentNotify {
  unsigned Msg;
  Word Event;
  Word ChildID;
  HWnd ChildWnd;
  Word Value;
  Smallint XPos;
  Smallint YPos;
  Word Value1;
  Longint Value2;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMParentNotify.

35.2.78 Messages.TWMPower Record

Pascal

```
TWMPower = packed record
  Msg: Cardinal;
  PowerEvt: Longint;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMPower {
  unsigned Msg;
  Longint PowerEvt;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMPower.

35.2.79 Messages.TWMPrint Record

Pascal

```
TWMPrint = packed record
  Msg: Cardinal;
  DC: HDC;
  Flags: Cardinal;
  Result: Integer;
end;
```

C++

```
struct TWMPrint {
  unsigned Msg;
  HDC DC;
  unsigned Flags;
  int Result;
};
```

File

Messages

Description

This is record Messages.TWMPrint.

35.2.80 Messages.TWMQueryEndSession Record

Pascal

```
TWMQueryEndSession = packed record
  Msg: Cardinal;
  Source: Longint;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMQueryEndSession {
  unsigned Msg;
  Longint Source;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMQueryEndSession.

35.2.81 Messages.TWMQuit Record

Pascal

```
TWMQuit = packed record
  Msg: Cardinal;
  ExitCode: Longint;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMQuit {
  unsigned Msg;
  Longint ExitCode;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMQuit.

35.2.82 Messages.TWMRenderFormat Record

Pascal

```
TWMRenderFormat = packed record
  Msg: Cardinal;
  Format: Longint;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMRenderFormat {
  unsigned Msg;
  Longint Format;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMRenderFormat.

35.2.83 Messages.TWMScroll Record

Pascal

```
TWMScroll = packed record
  Msg: Cardinal;
  ScrollCode: Smallint;
  Pos: Smallint;
  ScrollBar: HWND;
  Result: Longint;
end;
```

C++

```
struct TWMScroll {
  unsigned Msg;
  Smallint ScrollCode;
  Smallint Pos;
  HWND ScrollBar;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMScroll.

35.2.84 Messages.TWMSetCursor Record

Pascal

```
TWMSetCursor = packed record
  Msg: Cardinal;
  CursorWnd: HWND;
  HitTest: SmallInt;
  MouseMsg: Word;
  Result: Longint;
end;
```

C++

```
struct TWMSetCursor {
  unsigned Msg;
  HWND CursorWnd;
  SmallInt HitTest;
  Word MouseMsg;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSetCursor.

35.2.85 Messages.TWMSetFocus Record

Pascal

```
TWMSetFocus = packed record
  Msg: Cardinal;
  FocusedWnd: HWND;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMSetFocus {
  unsigned Msg;
  HWND FocusedWnd;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSetFocus.

35.2.86 Messages.TWMSetFont Record

Pascal

```
TWMSetFont = packed record
  Msg: Cardinal;
  Font: HFONT;
  Redraw: WordBool;
  Unused: Word;
  Result: Longint;
end;
```

C++

```
struct TWMSetFont {
  unsigned Msg;
  HFONT Font;
  WordBool Redraw;
  Word Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSetFont.

35.2.87 Messages.TWMSetHotKey Record

Pascal

```
TWMSetHotKey = packed record
  Msg: Cardinal;
  Key: Longint;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMSetHotKey {
  unsigned Msg;
  Longint Key;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSetHotKey.

35.2.88 Messages.TWMSetIcon Record

Pascal

```
TWMSetIcon = packed record
  Msg: Cardinal;
  BigIcon: Longbool;
  Icon: HICON;
  Result: Longint;
end;
```

C++

```
struct TWMSetIcon {
  unsigned Msg;
  Longbool BigIcon;
  HICON Icon;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSetIcon.

35.2.89 Messages.TWMSetRedraw Record

Pascal

```
TWMSetRedraw = packed record
  Msg: Cardinal;
  Redraw: Longint;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMSetRedraw {
  unsigned Msg;
  Longint Redraw;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSetRedraw.

35.2.90 Messages.TWMSetText Record

Pascal

```
TWMSetText = packed record
  Msg: Cardinal;
  Unused: Longint;
  Text: PChar;
  Result: Longint;
end;
```

C++

```
struct TWMSetText {
  unsigned Msg;
  Longint Unused;
  const char * Text;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSetText.

35.2.91 Messages.TWMSettingChange Record

Pascal

```
TWMSettingChange = packed record
  Msg: Cardinal;
  Flag: Integer;
  Section: PChar;
  Result: Longint;
end;
```

C++

```
struct TWMSettingChange {
  unsigned Msg;
  int Flag;
  const char * Section;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSettingChange.

35.2.92 Messages.TWMShowWindow Record

Pascal

```
TWMShowWindow = packed record
  Msg: Cardinal;
  Show: BOOL;
  Status: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMShowWindow {
  unsigned Msg;
  BOOL Show;
  Longint Status;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMShowWindow.

35.2.93 Messages.TWMSize Record

Pascal

```
TWMSize = packed record
  Msg: Cardinal;
  SizeType: Longint;
  Width: Word;
  Height: Word;
  Result: Longint;
end;
```

C++

```
struct TWMSize {
  unsigned Msg;
  Longint SizeType;
  Word Width;
  Word Height;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSize.

35.2.94 Messages.TWMSizeClipboard Record

Pascal

```
TWMSizeClipboard = packed record
  Msg: Cardinal;
  Viewer: HWND;
  RC: THandle;
  Result: Longint;
end;
```

C++

```
struct TWMSizeClipboard {
  unsigned Msg;
  HWND Viewer;
  THandle RC;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSizeClipboard.

35.2.95 Messages.TWMSpoolerStatus Record

Pascal

```
TWMSpoolerStatus = packed record
  Msg: Cardinal;
  JobStatus: Longint;
  JobsLeft: Word;
  Unused: Word;
  Result: Longint;
end;
```

C++

```
struct TWMSpoolerStatus {
  unsigned Msg;
  Longint JobStatus;
  Word JobsLeft;
  Word Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSpoolerStatus.

35.2.96 Messages.TWMStyleChange Record

Pascal

```
TWMStyleChange = packed record
  Msg: Cardinal;
  StyleType: Longint;
  StyleStruct: PStyleStruct;
  Result: Longint;
end;
```

C++

```
struct TWMStyleChange {
  unsigned Msg;
  Longint StyleType;
  PStyleStruct StyleStruct;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMStyleChange.

35.2.97 Messages.TWMSysCommand Record

Pascal

```
TWMSysCommand = packed record
  Msg: Cardinal;
  CmdType: Longint;
  case Longint of
    SC_HOTKEY: (ActivateWnd: HWND;);
    SC_KEYMENU: (Key: Word;);
    SC_VSCROLL: (XPos: Smallint;
                 YPos: Smallint;
                 Result: Longint;);
end;
```

C++

```
struct TWMSysCommand {
  unsigned Msg;
  Longint CmdType;
  HWND ActivateWnd;
  Word Key;
  Smallint XPos;
  Smallint YPos;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSysCommand.

35.2.98 Messages.TWMSysDeadChar Record

Pascal

```
TWMSysDeadChar = packed record
  Msg: Cardinal;
  CharCode: Word;
  Unused: Word;
  KeyData: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMSysDeadChar {
  unsigned Msg;
  Word CharCode;
  Word Unused;
  Longint KeyData;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSysDeadChar.

35.2.99 Messages.TWMSystemError Record

Pascal

```
TWMSystemError = packed record
  Msg: Cardinal;
  ErrSpec: Word;
  Unused: Longint;
  Result: Longint;
end;
```

C++

```
struct TWMSystemError {
  unsigned Msg;
  Word ErrSpec;
  Longint Unused;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMSystemError.

35.2.100 Messages.TWMTimer Record

Pascal

```
TWMTimer = packed record
  Msg: Cardinal;
  TimerID: Longint;
  TimerProc: TFarProc;
  Result: Longint;
end;
```

C++

```
struct TWMTimer {
  unsigned Msg;
  Longint TimerID;
  TFarProc TimerProc;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMTimer.

35.2.101 Messages.TWMUIState Record

Pascal

```
TWMUIState = packed record
  Msg: Cardinal;
  Action: Word;
  Flags: Word;
  Unused: Longint;
end;
```

C++

```
struct TWMUIState {
  unsigned Msg;
  Word Action;
  Word Flags;
  Longint Unused;
};
```

File

Messages

Description

This is record Messages.TWMUIState.

35.2.102 Messages.TWMVScrollClipboard Record

Pascal

```
TWMVScrollClipboard = packed record
  Msg: Cardinal;
  Viewer: HWND;
  ScollCode: Word;
  ThumbPos: Word;
  Result: Longint;
end;
```

C++

```
struct TWMVScrollClipboard {
  unsigned Msg;
  HWND Viewer;
  Word ScollCode;
  Word ThumbPos;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMVScrollClipboard.

35.2.103 Messages.TWMWindowPosMsg Record

Pascal

```
TWMWindowPosMsg = packed record
  Msg: Cardinal;
  Unused: Integer;
  WindowPos: PWindowPos;
  Result: Longint;
end;
```

C++

```
struct TWMWindowPosMsg {
  unsigned Msg;
  int Unused;
  PWindowPos WindowPos;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMWindowPosMsg.

35.2.104 Messages.TWMWinIniChange Record

Pascal

```
TWMWinIniChange = packed record
  Msg: Cardinal;
  Unused: Integer;
  Section: PChar;
  Result: Longint;
end;
```

C++

```
struct TWMWinIniChange {
  unsigned Msg;
  int Unused;
  const char * Section;
  Longint Result;
};
```

File

Messages

Description

This is record Messages.TWMWinIniChange.

35.3 Types

The following table lists types in this documentation.

Types

Name	Description
PMessage (see page 1569)	This is type Messages.PMessage.
TWMCancelMode (see page 1570)	This is type Messages.TWMCancelMode.
TWMChangeUIState (see page 1570)	This is type Messages.TWMChangeUIState.
TWMChar (see page 1570)	This is type Messages.TWMChar.
TWMChildActivate (see page 1571)	This is type Messages.TWMChildActivate.
TWMClear (see page 1571)	This is type Messages.TWMClear.
TWMClose (see page 1571)	This is type Messages.TWMClose.
TWMCopy (see page 1572)	This is type Messages.TWMCopy.
TWMCtlColorBtn (see page 1572)	This is type Messages.TWMCtlColorBtn.
TWMCtlColorDlg (see page 1572)	This is type Messages.TWMCtlColorDlg.
TWMCtlColorEdit (see page 1573)	This is type Messages.TWMCtlColorEdit.
TWMCtlColorListbox (see page 1573)	This is type Messages.TWMCtlColorListbox.
TWMCtlColorMsgbox (see page 1573)	This is type Messages.TWMCtlColorMsgbox.
TWMCtlColorScrollbar (see page 1574)	This is type Messages.TWMCtlColorScrollbar.
TWMCtlColorStatic (see page 1574)	This is type Messages.TWMCtlColorStatic.
TWMCut (see page 1574)	This is type Messages.TWMCut.
TWMDeadChar (see page 1575)	This is type Messages.TWMDeadChar.
TWMDestroy (see page 1575)	This is type Messages.TWMDestroy.
TWMDestroyClipboard (see page 1575)	This is type Messages.TWMDestroyClipboard.
TWMDrawClipboard (see page 1576)	This is type Messages.TWMDrawClipboard.
TWMExitMenuLoop (see page 1576)	This is type Messages.TWMExitMenuLoop.
TWMFontChange (see page 1576)	This is type Messages.TWMFontChange.
TWMGetDlgCode (see page 1577)	This is type Messages.TWMGetDlgCode.
TWMGetFont (see page 1577)	This is type Messages.TWMGetFont.
TWMGetHotKey (see page 1577)	This is type Messages.TWMGetHotKey.
TWMGetTextLength (see page 1578)	This is type Messages.TWMGetTextLength.
TWMHScroll (see page 1578)	This is type Messages.TWMHScroll.
TWMIconEraseBkgnd (see page 1578)	This is type Messages.TWMIconEraseBkgnd.
TWMKeyDown (see page 1579)	This is type Messages.TWMKeyDown.
TWMKeyUp (see page 1579)	This is type Messages.TWMKeyUp.
TWMLButtonDblClk (see page 1579)	This is type Messages.TWMLButtonDblClk.
TWMLButtonDown (see page 1580)	TWMLButtonDown represents the Windows message that occurs when the user clicks the left mouse button.
TWMLButtonUp (see page 1580)	This is type Messages.TWMLButtonUp.
TWMMButtonDblClk (see page 1580)	This is type Messages.TWMMButtonDblClk.
TWMMButtonDown (see page 1581)	This is type Messages.TWMMButtonDown.
TWMMButtonUp (see page 1581)	This is type Messages.TWMMButtonUp.
TWMMDIGetActive (see page 1581)	This is type Messages.TWMMDIGetActive.
TWMMDIIconArrange (see page 1582)	This is type Messages.TWMMDIIconArrange.
TWMMDIRefreshMenu (see page 1582)	This is type Messages.TWMMDIRefreshMenu.
TWMMouseMove (see page 1582)	This is type Messages.TWMMouseMove.
TWMNCDestroy (see page 1583)	This is type Messages.TWMNCDestroy.
TWMNCLButtonDblClk (see page 1583)	This is type Messages.TWMNCLButtonDblClk.
TWMNCLButtonDown (see page 1583)	This is type Messages.TWMNCLButtonDown.
TWMNCLButtonUp (see page 1584)	This is type Messages.TWMNCLButtonUp.

TWMNCMButtonDblClk (see page 1584)	This is type Messages.TWMNCMButtonDblClk.
TWMNCMButtonDown (see page 1584)	This is type Messages.TWMNCMButtonDown.
TWMNCMButtonUp (see page 1585)	This is type Messages.TWMNCMButtonUp.
TWMNCMouseMove (see page 1585)	This is type Messages.TWMNCMouseMove.
TWMNCRButtonDblClk (see page 1585)	This is type Messages.TWMNCRButtonDblClk.
TWMNCRButtonDown (see page 1586)	This is type Messages.TWMNCRButtonDown.
TWMNCRButtonUp (see page 1586)	This is type Messages.TWMNCRButtonUp.
TWMPaintIcon (see page 1586)	This is type Messages.TWMPaintIcon.
TWMPaste (see page 1587)	This is type Messages.TWMPaste.
TWMPrintClient (see page 1587)	This is type Messages.TWMPrintClient.
TWMQueryDragIcon (see page 1587)	This is type Messages.TWMQueryDragIcon.
TWMQueryNewPalette (see page 1588)	This is type Messages.TWMQueryNewPalette.
TWMQueryOpen (see page 1588)	This is type Messages.TWMQueryOpen.
TWMQueryUIState (see page 1588)	This is type Messages.TWMQueryUIState.
TWMQueueSync (see page 1589)	This is type Messages.TWMQueueSync.
TWMRButtonDblClk (see page 1589)	This is type Messages.TWMRButtonDblClk.
TWMRButtonDown (see page 1589)	This is type Messages.TWMRButtonDown.
TWMRButtonUp (see page 1590)	This is type Messages.TWMRButtonUp.
TWMRenderAllFormats (see page 1590)	This is type Messages.TWMRenderAllFormats.
TWMStyleChanged (see page 1590)	This is type Messages.TWMStyleChanged.
TWMStyleChanging (see page 1591)	This is type Messages.TWMStyleChanging.
TWMSysChar (see page 1591)	This is type Messages.TWMSysChar.
TWMSysColorChange (see page 1591)	This is type Messages.TWMSysColorChange.
TWMSysKeyDown (see page 1592)	This is type Messages.TWMSysKeyDown.
TWMSysKeyUp (see page 1592)	This is type Messages.TWMSysKeyUp.
TWMTimeChange (see page 1592)	This is type Messages.TWMTimeChange.
TWMUndo (see page 1593)	This is type Messages.TWMUndo.
TWMUpdateUIState (see page 1593)	This is type Messages.TWMUpdateUIState.
TWMVKeyToItem (see page 1593)	This is type Messages.TWMVKeyToItem.
TWMVScroll (see page 1594)	This is type Messages.TWMVScroll.
TWMWindowPosChanged (see page 1594)	This is type Messages.TWMWindowPosChanged.
TWMWindowPosChanging (see page 1594)	This is type Messages.TWMWindowPosChanging.

35.3.1 Messages.PMessage Type

Pascal

```
PMessage = ^TMessage;
```

C++

```
^TMessage PMessage;
```

File

Messages

Description

This is type Messages.PMessage.

35.3.2 Messages.TWMCancelMode Type

Pascal

```
TWMCancelMode = TWMNoParams;
```

C++

```
TWMNoParams TWMCancelMode;
```

File

Messages

Description

This is type Messages.TWMCancelMode.

35.3.3 Messages.TWMChangeUIState Type

Pascal

```
TWMChangeUIState = TWMUIState;
```

C++

```
TWMUIState TWMChangeUIState;
```

File

Messages

Description

This is type Messages.TWMChangeUIState.

35.3.4 Messages.TWMChar Type

Pascal

```
TWMChar = TWMKey;
```

C++

```
TWMKey TWMChar;
```

File

Messages

Description

This is type Messages.TWMChar.

35.3.5 Messages.TWMChildActivate Type

Pascal

```
TWMChildActivate = TWMNoParams;
```

C++

```
TWMNoParams TWMChildActivate;
```

File

Messages

Description

This is type Messages.TWMChildActivate.

35.3.6 Messages.TWMClear Type

Pascal

```
TWMClear = TWMNoParams;
```

C++

```
TWMNoParams TWMClear;
```

File

Messages

Description

This is type Messages.TWMClear.

35.3.7 Messages.TWMClose Type

Pascal

```
TWMClose = TWMNoParams;
```

C++

```
TWMNoParams TWMClose;
```

File

Messages

Description

This is type Messages.TWMClose.

35.3.8 Messages.TWMCopy Type

Pascal

```
TWMCopy = TWMNoParams;
```

C++

```
TWMNoParams TWMCopy;
```

File

Messages

Description

This is type Messages.TWMCopy.

35.3.9 Messages.TWMCtlColorBtn Type

Pascal

```
TWMCtlColorBtn = TWMCtlColor;
```

C++

```
TWMCtlColor TWMCtlColorBtn;
```

File

Messages

Description

This is type Messages.TWMCtlColorBtn.

35.3.10 Messages.TWMCtlColorDlg Type

Pascal

```
TWMCtlColorDlg = TWMCtlColor;
```

C++

```
TWMCtlColor TWMCtlColorDlg;
```

File

Messages

Description

This is type Messages.TWMCtlColorDlg.

35.3.11 Messages.TWMCtlColorEdit Type

Pascal

```
TWMCtlColorEdit = TWMCtlColor;
```

C++

```
TWMCtlColor TWMCtlColorEdit;
```

File

Messages

Description

This is type Messages.TWMCtlColorEdit.

35.3.12 Messages.TWMCtlColorListbox Type

Pascal

```
TWMCtlColorListbox = TWMCtlColor;
```

C++

```
TWMCtlColor TWMCtlColorListbox;
```

File

Messages

Description

This is type Messages.TWMCtlColorListbox.

35.3.13 Messages.TWMCtlColorMsgbox Type

Pascal

```
TWMCtlColorMsgbox = TWMCtlColor;
```

C++

```
TWMCtlColor TWMCtlColorMsgbox;
```

File

Messages

Description

This is type Messages.TWMCtlColorMsgbox.

35.3.14 Messages.TWMCtlColorScrollbar Type

Pascal

```
TWMCtlColorScrollbar = TWMCtlColor;
```

C++

```
TWMCtlColor TWMCtlColorScrollbar;
```

File

Messages

Description

This is type Messages.TWMCtlColorScrollbar.

35.3.15 Messages.TWMCtlColorStatic Type

Pascal

```
TWMCtlColorStatic = TWMCtlColor;
```

C++

```
TWMCtlColor TWMCtlColorStatic;
```

File

Messages

Description

This is type Messages.TWMCtlColorStatic.

35.3.16 Messages.TWMCut Type

Pascal

```
TWMCut = TWMNoParams;
```

C++

```
TWMNoParams TWMCut;
```

File

Messages

Description

This is type Messages.TWMCut.

35.3.17 Messages.TWMDeadChar Type

Pascal

```
TWMDeadChar = TWMChar;
```

C++

```
TWMChar TWMDeadChar;
```

File

Messages

Description

This is type Messages.TWMDeadChar.

35.3.18 Messages.TWMDestroy Type

Pascal

```
TWMDestroy = TWMNoParams;
```

C++

```
TWMNoParams TWMDestroy;
```

File

Messages

Description

This is type Messages.TWMDestroy.

35.3.19 Messages.TWMDestroyClipboard Type

Pascal

```
TWMDestroyClipboard = TWMNoParams;
```

C++

```
TWMNoParams TWMDestroyClipboard;
```

File

Messages

Description

This is type Messages.TWMDestroyClipboard.

35.3.20 Messages.TWMDrawClipboard Type

Pascal

```
TWMDrawClipboard = TWMNoParams;
```

C++

```
TWMNoParams TWMDrawClipboard;
```

File

Messages

Description

This is type Messages.TWMDrawClipboard.

35.3.21 Messages.TWMExitMenuLoop Type

Pascal

```
TWMExitMenuLoop = TWMEnterMenuLoop;
```

C++

```
TWMEnterMenuLoop TWMExitMenuLoop;
```

File

Messages

Description

This is type Messages.TWMExitMenuLoop.

35.3.22 Messages.TWMFontChange Type

Pascal

```
TWMFontChange = TWMNoParams;
```

C++

```
TWMNoParams TWMFontChange;
```

File

Messages

Description

This is type Messages.TWMFontChange.

35.3.23 Messages.TWMGetDlgCode Type

Pascal

```
TWMGetDlgCode = TWMNoParams;
```

C++

```
TWMNoParams TWMGetDlgCode;
```

File

Messages

Description

This is type Messages.TWMGetDlgCode.

35.3.24 Messages.TWMGetFont Type

Pascal

```
TWMGetFont = TWMNoParams;
```

C++

```
TWMNoParams TWMGetFont;
```

File

Messages

Description

This is type Messages.TWMGetFont.

35.3.25 Messages.TWMGetHotKey Type

Pascal

```
TWMGetHotKey = TWMNoParams;
```

C++

```
TWMNoParams TWMGetHotKey;
```

File

Messages

Description

This is type Messages.TWMGetHotKey.

35.3.26 Messages.TWMGetTextLength Type

Pascal

```
TWMGetTextLength = TWMNoParams;
```

C++

```
TWMNoParams TWMGetTextLength;
```

File

Messages

Description

This is type Messages.TWMGetTextLength.

35.3.27 Messages.TWMHScroll Type

Pascal

```
TWMHScroll = TWMScroll;
```

C++

```
TWMScroll TWMHScroll;
```

File

Messages

Description

This is type Messages.TWMHScroll.

35.3.28 Messages.TWMIconEraseBkgnd Type

Pascal

```
TWMIconEraseBkgnd = TWMEraseBkgnd;
```

C++

```
TWMEraseBkgnd TWMIconEraseBkgnd;
```

File

Messages

Description

This is type Messages.TWMIconEraseBkgnd.

35.3.29 Messages.TWMKeyDown Type

Pascal

```
TWMKeyDown = TWMKey;
```

C++

```
TWMKey TWMKeyDown;
```

File

Messages

Description

This is type Messages.TWMKeyDown.

35.3.30 Messages.TWMKeyUp Type

Pascal

```
TWMKeyUp = TWMKey;
```

C++

```
TWMKey TWMKeyUp;
```

File

Messages

Description

This is type Messages.TWMKeyUp.

35.3.31 Messages.TWMLButtonDblClk Type

Pascal

```
TWMLButtonDblClk = TWMMouse;
```

C++

```
TWMMouse TWMLButtonDblClk;
```

File

Messages

Description

This is type Messages.TWMLButtonDblClk.

35.3.32 Messages.TWMLButtonDown Type

TWMLButtonDown represents the Windows message that occurs when the user clicks the left mouse button.

Pascal

```
TWMLButtonDown = TWMMouse;
```

C++

```
TWMMouse TWMLButtonDown;
```

File

Messages

Description

TWMLButtonDown is the data structure that CLX uses for Windows messages informing an application that the user just clicked the left mouse button. It is the same as all mouse messages.

See Also

TMessage (see page 1513)

TWMMouse (see page 1543)

TWMPaint (see page 1551)

TWMMenuChar (see page 1541)

TWMNotify (see page 1550)

TWMKey (see page 1535)

35.3.33 Messages.TWMLButtonUp Type

Pascal

```
TWMLButtonUp = TWMMouse;
```

C++

```
TWMMouse TWMLButtonUp;
```

File

Messages

Description

This is type Messages.TWMLButtonUp.

35.3.34 Messages.TWMMButtonDblClk Type

Pascal

```
TWMMButtonDblClk = TWMMouse;
```

C++

```
TWMMouse TWMMButtonDblClk;
```

File

Messages

Description

This is type Messages.TWMMButtonDblClk.

35.3.35 Messages.TWMMButtonDown Type

Pascal

```
TWMMButtonDown = TWMMouse;
```

C++

```
TWMMouse TWMMButtonDown;
```

File

Messages

Description

This is type Messages.TWMMButtonDown.

35.3.36 Messages.TWMMButtonUp Type

Pascal

```
TWMMButtonUp = TWMMouse;
```

C++

```
TWMMouse TWMMButtonUp;
```

File

Messages

Description

This is type Messages.TWMMButtonUp.

35.3.37 Messages.TWMMDIGetActive Type

Pascal

```
TWMMDIGetActive = TWMNoParams;
```

C++

```
TWMNoParams TWMMDIGetActive;
```

File

Messages

Description

This is type Messages.TWMMDIGetActive.

35.3.38 Messages.TWMMDIIconArrange Type

Pascal

```
TWMMDIIconArrange = TWMNoParams;
```

C++

```
TWMNoParams TWMMDIIconArrange;
```

File

Messages

Description

This is type Messages.TWMMDIIconArrange.

35.3.39 Messages.TWMMDIRefreshMenu Type

Pascal

```
TWMMDIRefreshMenu = TWMNoParams;
```

C++

```
TWMNoParams TWMMDIRefreshMenu;
```

File

Messages

Description

This is type Messages.TWMMDIRefreshMenu.

35.3.40 Messages.TWMMouseMove Type

Pascal

```
TWMMouseMove = TWMMouse;
```

C++

```
TWMMouse TWMMouseMove;
```

File

Messages

Description

This is type Messages.TWMMouseMove.

35.3.41 Messages.TWMNCDestroy Type

Pascal

```
TWMNCDestroy = TWMNoParams;
```

C++

```
TWMNoParams TWMNCDestroy;
```

File

Messages

Description

This is type Messages.TWMNCDestroy.

35.3.42 Messages.TWMNCLButtonDblClk Type

Pascal

```
TWMNCLButtonDblClk = TWMNCHitMessage;
```

C++

```
TWMNCHitMessage TWMNCLButtonDblClk;
```

File

Messages

Description

This is type Messages.TWMNCLButtonDblClk.

35.3.43 Messages.TWMNCLButtonDown Type

Pascal

```
TWMNCLButtonDown = TWMNCHitMessage;
```

C++

```
TWMNCHitMessage TWMNCLButtonDown;
```

File

Messages

Description

This is type Messages.TWMNCLButtonDown.

35.3.44 Messages.TWMNCLButtonUp Type

Pascal

```
TWMNCLButtonUp = TWMNCHitMessage;
```

C++

```
TWMNCHitMessage TWMNCLButtonUp;
```

File

Messages

Description

This is type Messages.TWMNCLButtonUp.

35.3.45 Messages.TWMNCMButtonDblClk Type

Pascal

```
TWMNCMButtonDblClk = TWMNCHitMessage;
```

C++

```
TWMNCHitMessage TWMNCMButtonDblClk;
```

File

Messages

Description

This is type Messages.TWMNCMButtonDblClk.

35.3.46 Messages.TWMNCMButtonDown Type

Pascal

```
TWMNCMButtonDown = TWMNCHitMessage;
```

C++

```
TWMNCHitMessage TWMNCMButtonDown;
```

File

Messages

Description

This is type Messages.TWMNCMButtonDown.

35.3.47 Messages.TWMNCMButtonUp Type

Pascal

```
TWMNCMButtonUp = TWMNCHitMessage;
```

C++

```
TWMNCHitMessage TWMNCMButtonUp;
```

File

Messages

Description

This is type Messages.TWMNCMButtonUp.

35.3.48 Messages.TWMNCMouseMove Type

Pascal

```
TWMNCMouseMove = TWMNCHitMessage;
```

C++

```
TWMNCHitMessage TWMNCMouseMove;
```

File

Messages

Description

This is type Messages.TWMNCMouseMove.

35.3.49 Messages.TWMNCRButtonDblClk Type

Pascal

```
TWMNCRButtonDblClk = TWMNCHitMessage;
```

C++

```
TWMNCHitMessage TWMNCRButtonDblClk;
```

File

Messages

Description

This is type Messages.TWMNCRButtonDblClk.

35.3.50 Messages.TWMNCRButtonDown Type

Pascal

```
TWMNCRButtonDown = TWMNCHitMessage;
```

C++

```
TWMNCHitMessage TWMNCRButtonDown;
```

File

Messages

Description

This is type Messages.TWMNCRButtonDown.

35.3.51 Messages.TWMNCRButtonUp Type

Pascal

```
TWMNCRButtonUp = TWMNCHitMessage;
```

C++

```
TWMNCHitMessage TWMNCRButtonUp;
```

File

Messages

Description

This is type Messages.TWMNCRButtonUp.

35.3.52 Messages.TWMPaintIcon Type

Pascal

```
TWMPaintIcon = TWMNoParams;
```

C++

```
TWMNoParams TWMPaintIcon;
```

File

Messages

Description

This is type Messages.TWMPaintIcon.

35.3.53 Messages.TWMPaste Type

Pascal

```
TWMPaste = TWMNoParams;
```

C++

```
TWMNoParams TWMPaste;
```

File

Messages

Description

This is type Messages.TWMPaste.

35.3.54 Messages.TWMPrintClient Type

Pascal

```
TWMPrintClient = TWMPrint;
```

C++

```
TWMPrint TWMPrintClient;
```

File

Messages

Description

This is type Messages.TWMPrintClient.

35.3.55 Messages.TWMQueryDragIcon Type

Pascal

```
TWMQueryDragIcon = TWMNoParams;
```

C++

```
TWMNoParams TWMQueryDragIcon;
```

File

Messages

Description

This is type Messages.TWMQueryDragIcon.

35.3.56 Messages.TWMQueryNewPalette Type

Pascal

```
TWMQueryNewPalette = TWMNoParams;
```

C++

```
TWMNoParams TWMQueryNewPalette;
```

File

Messages

Description

This is type Messages.TWMQueryNewPalette.

35.3.57 Messages.TWMQueryOpen Type

Pascal

```
TWMQueryOpen = TWMNoParams;
```

C++

```
TWMNoParams TWMQueryOpen;
```

File

Messages

Description

This is type Messages.TWMQueryOpen.

35.3.58 Messages.TWMQueryUIState Type

Pascal

```
TWMQueryUIState = TWMNoParams;
```

C++

```
TWMNoParams TWMQueryUIState;
```

File

Messages

Description

This is type Messages.TWMQueryUIState.

35.3.59 Messages.TWMQueueSync Type

Pascal

```
TWMQueueSync = TWMNoParams;
```

C++

```
TWMNoParams TWMQueueSync;
```

File

Messages

Description

This is type Messages.TWMQueueSync.

35.3.60 Messages.TWMRButtonDblClk Type

Pascal

```
TWMRButtonDblClk = TWMMouse;
```

C++

```
TWMMouse TWMRButtonDblClk;
```

File

Messages

Description

This is type Messages.TWMRButtonDblClk.

35.3.61 Messages.TWMRButtonDown Type

Pascal

```
TWMRButtonDown = TWMMouse;
```

C++

```
TWMMouse TWMRButtonDown;
```

File

Messages

Description

This is type Messages.TWMRButtonDown.

35.3.62 Messages.TWMRButtonUp Type

Pascal

```
TWMRButtonUp = TWMMouse;
```

C++

```
TWMMouse TWMRButtonUp;
```

File

Messages

Description

This is type Messages.TWMRButtonUp.

35.3.63 Messages.TWMRenderAllFormats Type

Pascal

```
TWMRenderAllFormats = TWMNoParams;
```

C++

```
TWMNoParams TWMRenderAllFormats;
```

File

Messages

Description

This is type Messages.TWMRenderAllFormats.

35.3.64 Messages.TWMStyleChanged Type

Pascal

```
TWMStyleChanged = TWMStyleChange;
```

C++

```
TWMStyleChange TWMStyleChanged;
```

File

Messages

Description

This is type Messages.TWMStyleChanged.

35.3.65 Messages.TWMStyleChanging Type

Pascal

```
TWMStyleChanging = TWMStyleChange;
```

C++

```
TWMStyleChange TWMStyleChanging;
```

File

Messages

Description

This is type Messages.TWMStyleChanging.

35.3.66 Messages.TWMSysChar Type

Pascal

```
TWMSysChar = TWMKey;
```

C++

```
TWMKey TWMSysChar;
```

File

Messages

Description

This is type Messages.TWMSysChar.

35.3.67 Messages.TWMSysColorChange Type

Pascal

```
TWMSysColorChange = TWMNoParams;
```

C++

```
TWMNoParams TWMSysColorChange;
```

File

Messages

Description

This is type Messages.TWMSysColorChange.

35.3.68 Messages.TWMSysKeyDown Type

Pascal

```
TWMSysKeyDown = TWMKey;
```

C++

```
TWMKey TWMSysKeyDown;
```

File

Messages

Description

This is type Messages.TWMSysKeyDown.

35.3.69 Messages.TWMSysKeyUp Type

Pascal

```
TWMSysKeyUp = TWMKey;
```

C++

```
TWMKey TWMSysKeyUp;
```

File

Messages

Description

This is type Messages.TWMSysKeyUp.

35.3.70 Messages.TWMTimeChange Type

Pascal

```
TWMTimeChange = TWMNoParams;
```

C++

```
TWMNoParams TWMTimeChange;
```

File

Messages

Description

This is type Messages.TWMTimeChange.

35.3.71 Messages.TWMUndo Type

Pascal

```
TWMUndo = TWMNoParams;
```

C++

```
TWMNoParams TWMUndo;
```

File

Messages

Description

This is type Messages.TWMUndo.

35.3.72 Messages.TWMUpdateUIState Type

Pascal

```
TWMUpdateUIState = TWMUIState;
```

C++

```
TWMUIState TWMUpdateUIState;
```

File

Messages

Description

This is type Messages.TWMUpdateUIState.

35.3.73 Messages.TWMVKeyToItem Type

Pascal

```
TWMVKeyToItem = TWMCharToItem;
```

C++

```
TWMCharToItem TWMVKeyToItem;
```

File

Messages

Description

This is type Messages.TWMVKeyToItem.

35.3.74 Messages.TWMVScroll Type

Pascal

```
TWMVScroll = TWMScroll;
```

C++

```
TWMScroll TWMVScroll;
```

File

Messages

Description

This is type Messages.TWMVScroll.

35.3.75 Messages.TWMWindowPosChanged Type

Pascal

```
TWMWindowPosChanged = TWMWindowPosMsg;
```

C++

```
TWMWindowPosMsg TWMWindowPosChanged;
```

File

Messages

Description

This is type Messages.TWMWindowPosChanged.

35.3.76 Messages.TWMWindowPosChanging Type

Pascal

```
TWMWindowPosChanging = TWMWindowPosMsg;
```

C++

```
TWMWindowPosMsg TWMWindowPosChanging;
```

File

Messages

Description

This is type Messages.TWMWindowPosChanging.

35.4 Constants

The following table lists constants in this documentation.

Constants

Name	Description
BM_CLICK (see page 1602)	Represents the Windows message BM_CLICK.
BM_GETCHECK (see page 1603)	Represents the Windows message BM_GETCHECK.
BM_GETIMAGE (see page 1603)	Represents the Windows message BM_GETIMAGE.
BM_GETSTATE (see page 1604)	Represents the Windows message BM_GETSTATE.
BM_SETCHECK (see page 1604)	Represents the Windows message BM_SETCHECK.
BM_SETIMAGE (see page 1604)	Represents the Windows message BM_SETIMAGE.
BM_SETSTATE (see page 1605)	Represents the Windows message BM_SETSTATE.
BM_SETSTYLE (see page 1605)	Represents the Windows message BM_SETSTYLE.
BN_CLICKED (see page 1606)	Represents the Windows message BN_CLICKED.
BN_DBLCLK (see page 1606)	Represents the Windows message BN_DBLCLK.
BN_DISABLE (see page 1607)	Represents the Windows message BN_DISABLE.
BN_DOUBLECLICKED (see page 1607)	Represents the Windows message BN_DOUBLECLICKED.
BN_HILITE (see page 1607)	Represents the Windows message BN_HILITE.
BN_KILLFOCUS (see page 1608)	Represents the Windows message BN_KILLFOCUS.
BN_PAINT (see page 1608)	Represents the Windows message BN_PAINT.
BN_PUSHED (see page 1609)	Represents the Windows message BN_PUSHED.
BN_SETFOCUS (see page 1609)	Represents the Windows message BN_SETFOCUS.
BN_UNHILITE (see page 1610)	Represents the Windows message BN_UNHILITE.
BN_UNPUSHED (see page 1610)	Represents the Windows message BN_UNPUSHED.
CB_ADDSTRING (see page 1610)	Represents the Windows message CB_ADDSTRING.
CB_DELETESTRING (see page 1611)	Represents the Windows message CB_DELETESTRING.
CB_DIR (see page 1611)	Represents the Windows message CB_DIR.
CB_FINDSTRING (see page 1612)	Represents the Windows message CB_FINDSTRING.
CB_FINDSTRINGEXACT (see page 1612)	Represents the Windows message CB_FINDSTRINGEXACT.
CB_GETCOUNT (see page 1613)	Represents the Windows message CB_GETCOUNT.
CB_GETCURSEL (see page 1613)	Represents the Windows message CB_GETCURSEL.
CB_GETDROPPEDCONTROLRECT (see page 1613)	Represents the Windows message CB_GETDROPPEDCONTROLRECT.
CB_GETDROPPEDSTATE (see page 1614)	Represents the Windows message CB_GETDROPPEDSTATE.
CB_GETDROPPEDWIDTH (see page 1614)	Represents the Windows message CB_GETDROPPEDWIDTH.
CB_GETEDITSEL (see page 1615)	Represents the Windows message CB_GETEDITSEL.
CB_GETEXTENDEDUI (see page 1615)	Represents the Windows message CB_GETEXTENDEDUI.
CB_GETHORIZONTALEXTENT (see page 1616)	Represents the Windows message CB_GETHORIZONTALEXTENT.
CB_GETITEMDATA (see page 1616)	Represents the Windows message CB_GETITEMDATA.
CB_GETITEMHEIGHT (see page 1617)	Represents the Windows message CB_GETITEMHEIGHT.
CB_GETLBTEXT (see page 1617)	Represents the Windows message CB_GETLBTEXT.
CB_GETLBTEXTLEN (see page 1617)	Represents the Windows message CB_GETLBTEXTLEN.
CB_GETLOCALE (see page 1618)	Represents the Windows message CB_GETLOCALE.
CB_GETTOPINDEX (see page 1618)	Represents the Windows message CB_GETTOPINDEX.
CB_INITSTORAGE (see page 1619)	Represents the Windows message CB_INITSTORAGE.
CB_INSERTSTRING (see page 1619)	Represents the Windows message CB_INSERTSTRING.
CB_LIMITTEXT (see page 1620)	Represents the Windows message CB_LIMITTEXT.
CB_MSGMAX (see page 1620)	Represents the Windows message CB_MSGMAX.
CB_RESETCONTENT (see page 1620)	Represents the Windows message CB_RESETCONTENT.
CB_SELECTSTRING (see page 1621)	Represents the Windows message CB_SELECTSTRING.

CB_SETCURSEL (see page 1621)	Represents the Windows message CB_SETCURSEL.
CB_SETDROPPEDWIDTH (see page 1622)	Represents the Windows message CB_SETDROPPEDWIDTH.
CB_SETEDITSEL (see page 1622)	Represents the Windows message CB_SETEDITSEL.
CB_SETEXTENDEDUI (see page 1623)	Represents the Windows message CB_SETEXTENDEDUI.
CB_SETHORIZONTALEXTENT (see page 1623)	Represents the Windows message CB_SETHORIZONTALEXTENT.
CB_SETITEMDATA (see page 1623)	Represents the Windows message CB_SETITEMDATA.
CB_SETITEMHEIGHT (see page 1624)	Represents the Windows message CB_SETITEMHEIGHT.
CB_SETLOCALE (see page 1624)	Represents the Windows message CB_SETLOCALE.
CB_SETTOPINDEX (see page 1625)	Represents the Windows message CB_SETTOPINDEX.
CB_SHOWDROPDOWN (see page 1625)	Represents the Windows message CB_SHOWDROPDOWN.
CBN_CLOSEUP (see page 1626)	Represents the Windows message CBN_CLOSEUP.
CBN_DBLCLK (see page 1626)	Represents the Windows message CBN_DBLCLK.
CBN_DROPDOWN (see page 1626)	Represents the Windows message CBN_DROPDOWN.
CBN_EDITCHANGE (see page 1627)	Represents the Windows message CBN_EDITCHANGE.
CBN_EDITUPDATE (see page 1627)	Represents the Windows message CBN_EDITUPDATE.
CBN_ERRSPACE (see page 1628)	Represents the Windows message CBN_ERRSPACE.
CBN_KILLFOCUS (see page 1628)	Represents the Windows message CBN_KILLFOCUS.
CBN_SELCHANGE (see page 1629)	Represents the Windows message CBN_SELCHANGE.
CBN_SELENDCANCEL (see page 1629)	Represents the Windows message CBN_SELENDCANCEL.
CBN_SELENDOK (see page 1629)	Represents the Windows message CBN_SELENDOK.
CBN_SETFOCUS (see page 1630)	Represents the Windows message CBN_SETFOCUS.
DM_GETDEFID (see page 1630)	Represents the Windows message DM_GETDEFID.
DM_REPOSITION (see page 1631)	Represents the Windows message DM_REPOSITION.
DM_SETDEFID (see page 1631)	Represents the Windows message DM_SETDEFID.
EM_CANUNDO (see page 1632)	Represents the Windows message EM_CANUNDO.
EM_CHARFROMPOS (see page 1632)	Represents the Windows message EM_CHARFROMPOS.
EM_EMPTYUNDOBUFFER (see page 1632)	Represents the Windows message EM_EMPTYUNDOBUFFER.
EM_FMTLINES (see page 1633)	Represents the Windows message EM_FMTLINES.
EM_GETFIRSTVISIBLELINE (see page 1633)	Represents the Windows message EM_GETFIRSTVISIBLELINE.
EM_GETHANDLE (see page 1634)	Represents the Windows message EM_GETHANDLE.
EM_GETIMESTATUS (see page 1634)	Represents the Windows message EM_GETIMESTATUS.
EM_GETLIMITTEXT (see page 1635)	Represents the Windows message EM_GETLIMITTEXT.
EM_GETLINE (see page 1635)	Represents the Windows message EM_GETLINE.
EM_GETLINECOUNT (see page 1635)	Represents the Windows message EM_GETLINECOUNT.
EM_GETMARGINS (see page 1636)	Represents the Windows message EM_GETMARGINS.
EM_GETMODIFY (see page 1636)	Represents the Windows message EM_GETMODIFY.
EM_GETPASSWORDCHAR (see page 1637)	Represents the Windows message EM_GETPASSWORDCHAR.
EM_GETRECT (see page 1637)	Represents the Windows message EM_GETRECT.
EM_GETSEL (see page 1638)	Represents the Windows message EM_GETSEL.
EM_GETTHUMB (see page 1638)	Represents the Windows message EM_GETTHUMB.
EM_GETWORDBREAKPROC (see page 1638)	Represents the Windows message EM_GETWORDBREAKPROC.
EM_LIMITTEXT (see page 1639)	Represents the Windows message EM_LIMITTEXT.
EM_LINEFROMCHAR (see page 1639)	Represents the Windows message EM_LINEFROMCHAR.
EM_LINEINDEX (see page 1640)	Represents the Windows message EM_LINEINDEX.
EM_LINELENGTH (see page 1640)	Represents the Windows message EM_LINELENGTH.
EM_LINESCROLL (see page 1641)	Represents the Windows message EM_LINESCROLL.
EM_POSFROMCHAR (see page 1641)	Represents the Windows message EM_POSFROMCHAR.
EM_REPLACESEL (see page 1641)	Represents the Windows message EM_REPLACESEL.
EM_SCROLL (see page 1642)	Represents the Windows message EM_SCROLL.
EM_SCROLLCARET (see page 1642)	Represents the Windows message EM_SCROLLCARET.
EM_SETHANDLE (see page 1643)	Represents the Windows message EM_SETHANDLE.
EM_SETIMESTATUS (see page 1643)	Represents the Windows message EM_SETIMESTATUS.
EM_SETLIMITTEXT (see page 1644)	Represents the Windows message EM_SETLIMITTEXT.
EM_SETMARGINS (see page 1644)	Represents the Windows message EM_SETMARGINS.

EM_SETMODIFY (see page 1644)	Represents the Windows message EM_SETMODIFY.
EM_SETPASSWORDCHAR (see page 1645)	Represents the Windows message EM_SETPASSWORDCHAR.
EM_SETREADONLY (see page 1645)	Represents the Windows message EM_SETREADONLY.
EM_SETRECT (see page 1646)	Represents the Windows message EM_SETRECT.
EM_SETRECTNP (see page 1646)	Represents the Windows message EM_SETRECTNP.
EM_SETSEL (see page 1647)	Represents the Windows message EM_SETSEL.
EM_SETTABSTOPS (see page 1647)	Represents the Windows message EM_SETTABSTOPS.
EM_SETWORDBREAKPROC (see page 1647)	Represents the Windows message EM_SETWORDBREAKPROC.
EM_UNDO (see page 1648)	Represents the Windows message EM_UNDO.
EN_CHANGE (see page 1648)	Represents the Windows message EN_CHANGE.
EN_ERRSPACE (see page 1649)	Represents the Windows message EN_ERRSPACE.
EN_HSCROLL (see page 1649)	Represents the Windows message EN_HSCROLL.
EN_KILLFOCUS (see page 1650)	Represents the Windows message EN_KILLFOCUS.
EN_MAXTEXT (see page 1650)	Represents the Windows message EN_MAXTEXT.
EN_SETFOCUS (see page 1650)	Represents the Windows message EN_SETFOCUS.
EN_UPDATE (see page 1651)	Represents the Windows message EN_UPDATE.
EN_VSCROLL (see page 1651)	Represents the Windows message EN_VSCROLL.
LB_ADDFILE (see page 1652)	Represents the Windows message LB_ADDFILE.
LB_ADDSTRING (see page 1652)	Represents the Windows message LB_ADDSTRING.
LB_DELETESTRING (see page 1653)	Represents the Windows message LB_DELETESTRING.
LB_DIR (see page 1653)	Represents the Windows message LB_DIR.
LB_FINDSTRING (see page 1653)	Represents the Windows message LB_FINDSTRING.
LB_FINDSTRINGEXACT (see page 1654)	Represents the Windows message LB_FINDSTRINGEXACT.
LB_GETANCHORINDEX (see page 1654)	Represents the Windows message LB_GETANCHORINDEX.
LB_GETCARETINDEX (see page 1655)	Represents the Windows message LB_GETCARETINDEX.
LB_GETCOUNT (see page 1655)	Represents the Windows message LB_GETCOUNT.
LB_GETCURSEL (see page 1656)	Represents the Windows message LB_GETCURSEL.
LB_GETHORIZONTALEXTENT (see page 1656)	Represents the Windows message LB_GETHORIZONTALEXTENT.
LB_GETITEMDATA (see page 1656)	Represents the Windows message LB_GETITEMDATA.
LB_GETITEMHEIGHT (see page 1657)	Represents the Windows message LB_GETITEMHEIGHT.
LB_GETITEMRECT (see page 1657)	Represents the Windows message LB_GETITEMRECT.
LB_GETLOCALE (see page 1658)	Represents the Windows message LB_GETLOCALE.
LB_GETSEL (see page 1658)	Represents the Windows message LB_GETSEL.
LB_GETSELCOUNT (see page 1659)	Represents the Windows message LB_GETSELCOUNT.
LB_GETSELITEMS (see page 1659)	Represents the Windows message LB_GETSELITEMS.
LB_GETTEXT (see page 1659)	Represents the Windows message LB_GETTEXT.
LB_GETTEXTLEN (see page 1660)	Represents the Windows message LB_GETTEXTLEN.
LB_GETTOPINDEX (see page 1660)	Represents the Windows message LB_GETTOPINDEX.
LB_INITSTORAGE (see page 1661)	Represents the Windows message LB_INITSTORAGE.
LB_INSERTSTRING (see page 1661)	Represents the Windows message LB_INSERTSTRING.
LB_ITEMFROMPOINT (see page 1662)	Represents the Windows message LB_ITEMFROMPOINT.
LB_MSGMAX (see page 1662)	Represents the Windows message LB_MSGMAX.
LB_RESETCONTENT (see page 1662)	Represents the Windows message LB_RESETCONTENT.
LB_SELECTSTRING (see page 1663)	Represents the Windows message LB_SELECTSTRING.
LB_SELITEMRANGE (see page 1663)	Represents the Windows message LB_SELITEMRANGE.
LB_SELITEMRANGEEX (see page 1664)	Represents the Windows message LB_SELITEMRANGEEX.
LB_SETANCHORINDEX (see page 1664)	Represents the Windows message LB_SETANCHORINDEX.
LB_SETCARETINDEX (see page 1665)	Represents the Windows message LB_SETCARETINDEX.
LB_SETCOLUMNWIDTH (see page 1665)	Represents the Windows message LB_SETCOLUMNWIDTH.
LB_SETCOUNT (see page 1665)	Represents the Windows message LB_SETCOUNT.
LB_SETCURSEL (see page 1666)	Represents the Windows message LB_SETCURSEL.
LB_SETHORIZONTALEXTENT (see page 1666)	Represents the Windows message LB_SETHORIZONTALEXTENT.
LB_SETITEMDATA (see page 1667)	Represents the Windows message LB_SETITEMDATA.
LB_SETITEMHEIGHT (see page 1667)	Represents the Windows message LB_SETITEMHEIGHT.

LB_SETLOCALE (see page 1668)	Represents the Windows message LB_SETLOCALE.
LB_SETSEL (see page 1668)	Represents the Windows message LB_SETSEL.
LB_SETTABSTOPS (see page 1668)	Represents the Windows message LB_SETTABSTOPS.
LB_SETTOPINDEX (see page 1669)	Represents the Windows message LB_SETTOPINDEX.
LBN_DBLCLK (see page 1669)	Represents the Windows message LBN_DBLCLK.
LBN_ERRSPACE (see page 1670)	Represents the Windows message LBN_ERRSPACE.
LBN_KILLFOCUS (see page 1670)	Represents the Windows message LBN_KILLFOCUS.
LBN_SELCANCEL (see page 1671)	Represents the Windows message LBN_SELCANCEL.
LBN_SELCHANGE (see page 1671)	Represents the Windows message LBN_SELCHANGE.
LBN_SETFOCUS (see page 1671)	Represents the Windows message LBN_SETFOCUS.
PSM_PAGEINFO (see page 1672)	Represents the Windows message PSM_PAGEINFO.
PSM_SHEETINFO (see page 1672)	Represents the Windows message PSM_SHEETINFO.
SBM_ENABLE_ARROWS (see page 1673)	Represents the Windows message SBM_ENABLE_ARROWS.
SBM_GETPOS (see page 1673)	Represents the Windows message SBM_GETPOS.
SBM_GETRANGE (see page 1674)	Represents the Windows message SBM_GETRANGE.
SBM_GETSCROLLBARINFO (see page 1674)	Represents the Windows message SBM_GETSCROLLBARINFO.
SBM_GETSCROLLINFO (see page 1674)	Represents the Windows message SBM_GETSCROLLINFO.
SBM_SETPOS (see page 1675)	Represents the Windows message SBM_SETPOS.
SBM_SETRANGE (see page 1675)	Represents the Windows message SBM_SETRANGE.
SBM_SETRANGEREDRAW (see page 1676)	Represents the Windows message SBM_SETRANGEREDRAW.
SBM_SETSCROLLINFO (see page 1676)	Represents the Windows message SBM_SETSCROLLINFO.
WM_ACTIVATE (see page 1677)	Represents the Windows message WM_ACTIVATE.
WM_ACTIVATEAPP (see page 1677)	Represents the Windows message WM_ACTIVATEAPP.
WM_APP (see page 1678)	Represents the Windows message WM_APP.
WM_APPCOMMAND (see page 1678)	Represents the Windows message WM_APPCOMMAND.
WM_ASKCBFORMATNAME (see page 1678)	Represents the Windows message WM_ASKCBFORMATNAME.
WM_CANCELJOURNAL (see page 1679)	Represents the Windows message WM_CANCELJOURNAL.
WM_CANCELMODE (see page 1679)	Represents the Windows message WM_CANCELMODE.
WM_CAPTURECHANGED (see page 1680)	Represents the Windows message WM_CAPTURECHANGED.
WM_CHANGECBCHAIN (see page 1680)	Represents the Windows message WM_CHANGECBCHAIN.
WM_CHANGEUISTATE (see page 1681)	Represents the Windows message WM_CHANGEUISTATE.
WM_CHAR (see page 1681)	Represents the Windows message WM_CHAR.
WM_CHARTOITEM (see page 1681)	Represents the Windows message WM_CHARTOITEM.
WM_CHILDACTIVATE (see page 1682)	Represents the Windows message WM_CHILDACTIVATE.
WM_CLEAR (see page 1682)	Represents the Windows message WM_CLEAR.
WM_CLOSE (see page 1683)	Represents the Windows message WM_CLOSE.
WM_COALESCE_FIRST (see page 1683)	Represents the Windows message WM_COALESCE_FIRST.
WM_COALESCE_LAST (see page 1684)	Represents the Windows message WM_COALESCE_LAST.
WM_COMMAND (see page 1684)	Represents the Windows message WM_COMMAND.
WM_COMMNOTIFY (see page 1685)	Represents the Windows message WM_COMMNOTIFY.
WM_COMPACTING (see page 1685)	Represents the Windows message WM_COMPACTING.
WM_COMPAREITEM (see page 1685)	Represents the Windows message WM_COMPAREITEM.
WM_CONTEXTMENU (see page 1686)	Represents the Windows message WM_CONTEXTMENU.
WM_COPY (see page 1686)	Represents the Windows message WM_COPY.
WM_COPYDATA (see page 1687)	Represents the Windows message WM_COPYDATA.
WM_CREATE (see page 1687)	Represents the Windows message WM_CREATE.
WM_CTLCOLOR (see page 1688)	Represents the Windows message WM_CTLCOLOR.
WM_CTLCOLORBTN (see page 1688)	Represents the Windows message WM_CTLCOLORBTN.
WM_CTLCOLORDLG (see page 1688)	Represents the Windows message WM_CTLCOLORDLG.
WM_CTLCOLOREDIT (see page 1689)	Represents the Windows message WM_CTLCOLOREDIT.
WM_CTLCOLORLISTBOX (see page 1689)	Represents the Windows message WM_CTLCOLORLISTBOX.
WM_CTLCOLORMSGBOX (see page 1690)	Represents the Windows message WM_CTLCOLORMSGBOX.
WM_CTLCOLORSCROLLBAR (see page 1690)	Represents the Windows message WM_CTLCOLORSCROLLBAR.
WM_CTLCOLORSTATIC (see page 1691)	Represents the Windows message WM_CTLCOLORSTATIC.

WM_CUT (see page 1691)	Represents the Windows message WM_CUT.
WM_DDE_ACK (see page 1691)	Represents the Windows message WM_DDE_ACK.
WM_DDE_ADVISE (see page 1692)	Represents the Windows message WM_DDE_ADVISE.
WM_DDE_DATA (see page 1692)	Represents the Windows message WM_DDE_DATA.
WM_DDE_EXECUTE (see page 1693)	Represents the Windows message WM_DDE_EXECUTE.
WM_DDE_FIRST (see page 1693)	Represents the Windows message WM_DDE_FIRST.
WM_DDE_INITIATE (see page 1694)	Represents the Windows message WM_DDE_INITIATE.
WM_DDE_LAST (see page 1694)	Represents the Windows message WM_DDE_LAST.
WM_DDE_POKE (see page 1695)	Represents the Windows message WM_DDE_POKE.
WM_DDE_REQUEST (see page 1695)	Represents the Windows message WM_DDE_REQUEST.
WM_DDE_TERMINATE (see page 1695)	Represents the Windows message WM_DDE_TERMINATE.
WM_DDE_UNADVISE (see page 1696)	Represents the Windows message WM_DDE_UNADVISE.
WM_DEADCHAR (see page 1696)	Represents the Windows message WM_DEADCHAR.
WM_DELETEITEM (see page 1697)	Represents the Windows message WM_DELETEITEM.
WM_DESTROY (see page 1697)	Represents the Windows message WM_DESTROY.
WM_DESTROYCLIPBOARD (see page 1698)	Represents the Windows message WM_DESTROYCLIPBOARD.
WM_DEVICECHANGE (see page 1698)	Represents the Windows message WM_DEVICECHANGE.
WM_DEVMODECHANGE (see page 1698)	Represents the Windows message WM_DEVMODECHANGE.
WM_DISPLAYCHANGE (see page 1699)	Represents the Windows message WM_DISPLAYCHANGE.
WM_DRAWCLIPBOARD (see page 1699)	Represents the Windows message WM_DRAWCLIPBOARD.
WM_DRAWITEM (see page 1700)	Represents the Windows message WM_DRAWITEM.
WM_DROPFILES (see page 1700)	Represents the Windows message WM_DROPFILES.
WM_DWMCOLORIZATIONCOLORCHANGED (see page 1701)	Represents the Windows message WM_DWMCOLORIZATIONCOLORCHANGED.
WM_DWMCOMPOSITIONCHANGED (see page 1701)	Represents the Windows message WM_DWMCOMPOSITIONCHANGED.
WM_DWMNCRENDERINGCHANGED (see page 1702)	Represents the Windows message WM_DWMNCRENDERINGCHANGED.
WM_DWMWINDOWMAXIMIZEDCHANGE (see page 1702)	Represents the Windows message WM_DWMWINDOWMAXIMIZEDCHANGE.
WM_ENABLE (see page 1703)	Represents the Windows message WM_ENABLE.
WM_ENDSESSION (see page 1703)	Represents the Windows message WM_ENDSESSION.
WM_ENTERIDLE (see page 1703)	Represents the Windows message WM_ENTERIDLE.
WM_ENTERMENULOOP (see page 1704)	Represents the Windows message WM_ENTERMENULOOP.
WM_ENTERSIZEMOVE (see page 1704)	Represents the Windows message WM_ENTERSIZEMOVE.
WM_ERASEBKGND (see page 1705)	Represents the Windows message WM_ERASEBKGND.
WM_EXITMENULOOP (see page 1705)	Represents the Windows message WM_EXITMENULOOP.
WM_EXITSIZEMOVE (see page 1706)	Represents the Windows message WM_EXITSIZEMOVE.
WM_FONTCHANGE (see page 1706)	Represents the Windows message WM_FONTCHANGE.
WM_GETDLGCODE (see page 1707)	Represents the Windows message WM_GETDLGCODE.
WM_GETFONT (see page 1707)	Represents the Windows message WM_GETFONT.
WM_GETHOTKEY (see page 1707)	Represents the Windows message WM_GETHOTKEY.
WM_GETICON (see page 1708)	Represents the Windows message WM_GETICON.
WM_GETMINMAXINFO (see page 1708)	Represents the Windows message WM_GETMINMAXINFO.
WM_GETOBJECT (see page 1709)	Represents the Windows message WM_GETOBJECT.
WM_GETTEXT (see page 1709)	Represents the Windows message WM_GETTEXT.
WM_GETTEXTLENGTH (see page 1710)	Represents the Windows message WM_GETTEXTLENGTH.
WM_HANDHELDFIRST (see page 1710)	Represents the Windows message WM_HANDHELDFIRST.
WM_HANDHELDLAST (see page 1711)	Represents the Windows message WM_HANDHELDLAST.
WM_HELP (see page 1711)	Represents the Windows message WM_HELP.
WM_HOTKEY (see page 1711)	Represents the Windows message WM_HOTKEY.
WM_HSCROLL (see page 1712)	Represents the Windows message WM_HSCROLL.
WM_HSCROLLCLIPBOARD (see page 1712)	Represents the Windows message WM_HSCROLLCLIPBOARD.
WM_ICONERASEBKGND (see page 1713)	Represents the Windows message WM_ICONERASEBKGND.
WM_IME_CHAR (see page 1713)	Represents the Windows message WM_IME_CHAR.
WM_IME_COMPOSITION (see page 1714)	Represents the Windows message WM_IME_COMPOSITION.
WM_IME_COMPOSITIONFULL (see page 1714)	Represents the Windows message WM_IME_COMPOSITIONFULL.
WM_IME_CONTROL (see page 1715)	Represents the Windows message WM_IME_CONTROL.

WM_IME_ENDCOMPOSITION (see page 1715)	Represents the Windows message WM_IME_ENDCOMPOSITION.
WM_IME_KEYDOWN (see page 1715)	Represents the Windows message WM_IME_KEYDOWN.
WM_IME_KEYLAST (see page 1716)	Represents the Windows message WM_IME_KEYLAST.
WM_IME_KEYUP (see page 1716)	Represents the Windows message WM_IME_KEYUP.
WM_IME_NOTIFY (see page 1717)	Represents the Windows message WM_IME_NOTIFY.
WM_IME_REQUEST (see page 1717)	Represents the Windows message WM_IME_REQUEST.
WM_IME_SELECT (see page 1718)	Represents the Windows message WM_IME_SELECT.
WM_IME_SETCONTEXT (see page 1718)	Represents the Windows message WM_IME_SETCONTEXT.
WM_IME_STARTCOMPOSITION (see page 1718)	Represents the Windows message WM_IME_STARTCOMPOSITION.
WM_INITDIALOG (see page 1719)	Represents the Windows message WM_INITDIALOG.
WM_INITMENU (see page 1719)	Represents the Windows message WM_INITMENU.
WM_INITMENUPOPUP (see page 1720)	Represents the Windows message WM_INITMENUPOPUP.
WM_INPUT (see page 1720)	Represents the Windows message WM_INPUT.
WM_INPUTLANGCHANGE (see page 1721)	Represents the Windows message WM_INPUTLANGCHANGE.
WM_INPUTLANGCHANGEREQUEST (see page 1721)	Represents the Windows message WM_INPUTLANGCHANGEREQUEST.
WM_KEYDOWN (see page 1722)	Represents the Windows message WM_KEYDOWN.
WM_KEYFIRST (see page 1722)	Represents the Windows message WM_KEYFIRST.
WM_KEYLAST (see page 1722)	Represents the Windows message WM_KEYLAST.
WM_KEYUP (see page 1723)	Represents the Windows message WM_KEYUP.
WM_KILLFOCUS (see page 1723)	Represents the Windows message WM_KILLFOCUS.
WM_LBUTTONDBLCLK (see page 1724)	Represents the Windows message WM_LBUTTONDBLCLK.
WM_LBUTTONDOWN (see page 1724)	Represents the Windows message WM_LBUTTONDOWN.
WM_LBUTTONUP (see page 1725)	Represents the Windows message WM_LBUTTONUP.
WM_MBUTTONDBLCLK (see page 1725)	Represents the Windows message WM_MBUTTONDBLCLK.
WM_MBUTTONDOWN (see page 1726)	Represents the Windows message WM_MBUTTONDOWN.
WM_MBUTTONUP (see page 1726)	Represents the Windows message WM_MBUTTONUP.
WM_MDIACTIVATE (see page 1726)	Represents the Windows message WM_MDIACTIVATE.
WM_MDICASCADE (see page 1727)	Represents the Windows message WM_MDICASCADE.
WM_MDICREATE (see page 1727)	Represents the Windows message WM_MDICREATE.
WM_MDIDESTROY (see page 1728)	Represents the Windows message WM_MDIDESTROY.
WM_MDIGETACTIVE (see page 1728)	Represents the Windows message WM_MDIGETACTIVE.
WM_MDIICONARRANGE (see page 1729)	Represents the Windows message WM_MDIICONARRANGE.
WM_MDIMAXIMIZE (see page 1729)	Represents the Windows message WM_MDIMAXIMIZE.
WM_MDINEXT (see page 1729)	Represents the Windows message WM_MDINEXT.
WM_MDIREFRESHMENU (see page 1730)	Represents the Windows message WM_MDIREFRESHMENU.
WM_MDIRESTORE (see page 1730)	Represents the Windows message WM_MDIRESTORE.
WM_MDISETMENU (see page 1731)	Represents the Windows message WM_MDISETMENU.
WM_MDITILE (see page 1731)	Represents the Windows message WM_MDITILE.
WM_MEASUREITEM (see page 1732)	Represents the Windows message WM_MEASUREITEM.
WM_MENUCHAR (see page 1732)	Represents the Windows message WM_MENUCHAR.
WM_MENUCOMMAND (see page 1732)	Represents the Windows message WM_MENUCOMMAND.
WM_MENUDRAG (see page 1733)	Represents the Windows message WM_MENUDRAG.
WM_MENUGETOBJECT (see page 1733)	Represents the Windows message WM_MENUGETOBJECT.
WM_MENURBUTTONUP (see page 1734)	Represents the Windows message WM_MENURBUTTONUP.
WM_MENUSELECT (see page 1734)	Represents the Windows message WM_MENUSELECT.
WM_MOUSEACTIVATE (see page 1735)	Represents the Windows message WM_MOUSEACTIVATE.
WM_MOUSEFIRST (see page 1735)	Represents the Windows message WM_MOUSEFIRST.
WM_MOUSEHOVER (see page 1736)	Represents the Windows message WM_MOUSEHOVER.
WM_MOUSELAST (see page 1736)	Represents the Windows message WM_MOUSELAST.
WM_MOUSELEAVE (see page 1736)	Represents the Windows message WM_MOUSELEAVE.
WM_MOUSEMOVE (see page 1737)	Represents the Windows message WM_MOUSEMOVE.
WM_MOUSEWHEEL (see page 1737)	Represents the Windows message WM_MOUSEWHEEL.
WM_MOVE (see page 1738)	Represents the Windows message WM_MOVE.
WM_MOVING (see page 1738)	Represents the Windows message WM_MOVING.

WM_NCACTIVATE (see page 1739)	Represents the Windows message WM_NCACTIVATE.
WM_NCCALCSIZE (see page 1739)	Represents the Windows message WM_NCCALCSIZE.
WM_NCCREATE (see page 1739)	Represents the Windows message WM_NCCREATE.
WM_NCDESTROY (see page 1740)	Represents the Windows message WM_NCDESTROY.
WM_NCHITTEST (see page 1740)	Represents the Windows message WM_NCHITTEST.
WM_NCLBUTTONDBLCLK (see page 1741)	Represents the Windows message WM_NCLBUTTONDBLCLK.
WM_NCLBUTTONDOWN (see page 1741)	Represents the Windows message WM_NCLBUTTONDOWN.
WM_NCLBUTTONUP (see page 1742)	Represents the Windows message WM_NCLBUTTONUP.
WM_NCMBUTTONDBLCLK (see page 1742)	Represents the Windows message WM_NCMBUTTONDBLCLK.
WM_NCMBUTTONDOWN (see page 1742)	Represents the Windows message WM_NCMBUTTONDOWN.
WM_NCMBUTTONUP (see page 1743)	Represents the Windows message WM_NCMBUTTONUP.
WM_NCMOUSEHOVER (see page 1743)	Represents the Windows message WM_NCMOUSEHOVER.
WM_NCMOUSELEAVE (see page 1744)	Represents the Windows message WM_NCMOUSELEAVE.
WM_NCMOUSEMOVE (see page 1744)	Represents the Windows message WM_NCMOUSEMOVE.
WM_NCPAINT (see page 1745)	Represents the Windows message WM_NCPAINT.
WM_NCRBUTTONDBLCLK (see page 1745)	Represents the Windows message WM_NCRBUTTONDBLCLK.
WM_NCRBUTTONDOWN (see page 1746)	Represents the Windows message WM_NCRBUTTONDOWN.
WM_NCRBUTTONUP (see page 1746)	Represents the Windows message WM_NCRBUTTONUP.
WM_NCXBUTTONDBLCLK (see page 1746)	Represents the Windows message WM_NCXBUTTONDBLCLK.
WM_NCXBUTTONDOWN (see page 1747)	Represents the Windows message WM_NCXBUTTONDOWN.
WM_NCXBUTTONUP (see page 1747)	Represents the Windows message WM_NCXBUTTONUP.
WM_NEXTDLGCTL (see page 1748)	Represents the Windows message WM_NEXTDLGCTL.
WM_NEXTMENU (see page 1748)	Represents the Windows message WM_NEXTMENU.
WM_NOTIFY (see page 1749)	Represents the Windows message WM_NOTIFY.
WM_NOTIFYFORMAT (see page 1749)	Represents the Windows message WM_NOTIFYFORMAT.
WM_NULL (see page 1749)	Represents the Windows message WM_NULL.
WM_PAINT (see page 1750)	Represents the Windows message WM_PAINT.
WM_PAINTCLIPBOARD (see page 1750)	Represents the Windows message WM_PAINTCLIPBOARD.
WM_PAINTICON (see page 1751)	Represents the Windows message WM_PAINTICON.
WM_PALETTECHANGED (see page 1751)	Represents the Windows message WM_PALETTECHANGED.
WM_PALETTEISCHANGING (see page 1752)	Represents the Windows message WM_PALETTEISCHANGING.
WM_PARENTNOTIFY (see page 1752)	Represents the Windows message WM_PARENTNOTIFY.
WM_PASTE (see page 1752)	Represents the Windows message WM_PASTE.
WM_PENWINFIRST (see page 1753)	Represents the Windows message WM_PENWINFIRST.
WM_PENWINLAST (see page 1753)	Represents the Windows message WM_PENWINLAST.
WM_POWER (see page 1754)	Represents the Windows message WM_POWER.
WM_POWERBROADCAST (see page 1754)	Represents the Windows message WM_POWERBROADCAST.
WM_PRINT (see page 1755)	Represents the Windows message WM_PRINT.
WM_PRINTCLIENT (see page 1755)	Represents the Windows message WM_PRINTCLIENT.
WM_QUERYDRAGICON (see page 1756)	Represents the Windows message WM_QUERYDRAGICON.
WM_QUERYENDSESSION (see page 1756)	Represents the Windows message WM_QUERYENDSESSION.
WM_QUERYNEWPALETTE (see page 1756)	Represents the Windows message WM_QUERYNEWPALETTE.
WM_QUERYOPEN (see page 1757)	Represents the Windows message WM_QUERYOPEN.
WM_QUERYUISTATE (see page 1757)	Represents the Windows message WM_QUERYUISTATE.
WM_QUEUESYNC (see page 1758)	Represents the Windows message WM_QUEUESYNC.
WM_QUIT (see page 1758)	Represents the Windows message WM_QUIT.
WM_RBUTTONDBLCLK (see page 1759)	Represents the Windows message WM_RBUTTONDBLCLK.
WM_RBUTTONDOWN (see page 1759)	Represents the Windows message WM_RBUTTONDOWN.
WM_RBUTTONUP (see page 1759)	Represents the Windows message WM_RBUTTONUP.
WM_RENDERALLFORMATS (see page 1760)	Represents the Windows message WM_RENDERALLFORMATS.
WM_RENDERFORMAT (see page 1760)	Represents the Windows message WM_RENDERFORMAT.
WM_SETCURSOR (see page 1761)	Represents the Windows message WM_SETCURSOR.
WM_SETFOCUS (see page 1761)	Represents the Windows message WM_SETFOCUS.
WM_SETFONT (see page 1762)	Represents the Windows message WM_SETFONT.

WM_SETHOTKEY (see page 1762)	Represents the Windows message WM_SETHOTKEY.
WM_SETICON (see page 1763)	Represents the Windows message WM_SETICON.
WM_SETREDRAW (see page 1763)	Represents the Windows message WM_SETREDRAW.
WM_SETTEXT (see page 1763)	Represents the Windows message WM_SETTEXT.
WM_SETTINGCHANGE (see page 1764)	Represents the Windows message WM_SETTINGCHANGE.
WM_SHOWWINDOW (see page 1764)	Represents the Windows message WM_SHOWWINDOW.
WM_SIZE (see page 1765)	Represents the Windows message WM_SIZE.
WM_SIZECLIPBOARD (see page 1765)	Represents the Windows message WM_SIZECLIPBOARD.
WM_SIZING (see page 1766)	Represents the Windows message WM_SIZING.
WM_SPOOLERSTATUS (see page 1766)	Represents the Windows message WM_SPOOLERSTATUS.
WM_STYLECHANGED (see page 1766)	Represents the Windows message WM_STYLECHANGED.
WM_STYLECHANGING (see page 1767)	Represents the Windows message WM_STYLECHANGING.
WM_SYSCHAR (see page 1767)	Represents the Windows message WM_SYSCHAR.
WM_SYSCOLORCHANGE (see page 1768)	Represents the Windows message WM_SYSCOLORCHANGE.
WM_SYSCOMMAND (see page 1768)	Represents the Windows message WM_SYSCOMMAND.
WM_SYSDEADCHAR (see page 1769)	Represents the Windows message WM_SYSDEADCHAR.
WM_SYSKEYDOWN (see page 1769)	Represents the Windows message WM_SYSKEYDOWN.
WM_SYSKEYUP (see page 1770)	Represents the Windows message WM_SYSKEYUP.
WM_SYSTEMERROR (see page 1770)	Represents the Windows message WM_SYSTEMERROR.
WM_TABLET_FIRST (see page 1770)	Represents the Windows message WM_TABLET_FIRST.
WM_TABLET_LAST (see page 1771)	Represents the Windows message WM_TABLET_LAST.
WM_TCARD (see page 1771)	Represents the Windows message WM_TCARD.
WM_THEMECHANGED (see page 1772)	Represents the Windows message WM_THEMECHANGED.
WM_TIMECHANGE (see page 1772)	Represents the Windows message WM_TIMECHANGE.
WM_TIMER (see page 1773)	Represents the Windows message WM_TIMER.
WM_UNDO (see page 1773)	Represents the Windows message WM_UNDO.
WM_UNICHAR (see page 1774)	Represents the Windows message WM_UNICHAR.
WM_UNINITMENUPOPUP (see page 1774)	Represents the Windows message WM_UNINITMENUPOPUP.
WM_UPDATEUISTATE (see page 1774)	Represents the Windows message WM_UPDATEUISTATE.
WM_USER (see page 1775)	Represents the Windows message WM_USER.
WM_USERCHANGED (see page 1775)	Represents the Windows message WM_USERCHANGED.
WM_VKEYTOITEM (see page 1776)	Represents the Windows message WM_VKEYTOITEM.
WM_VSCROLL (see page 1776)	Represents the Windows message WM_VSCROLL.
WM_VSCROLLCLIPBOARD (see page 1777)	Represents the Windows message WM_VSCROLLCLIPBOARD.
WM_WINDOWPOSCHANGED (see page 1777)	Represents the Windows message WM_WINDOWPOSCHANGED.
WM_WINDOWPOSCHANGING (see page 1778)	Represents the Windows message WM_WINDOWPOSCHANGING.
WM_WININICHANGE (see page 1778)	Represents the Windows message WM_WININICHANGE.
WM_WTSSESSION_CHANGE (see page 1778)	Represents the Windows message WM_WTSSESSION_CHANGE.

35.4.1 Messages.BM_CLICK Constant

Represents the Windows message BM_CLICK.

Pascal

```
BM_CLICK = $00F5;
```

C++

```
BM_CLICK = 0x00F5;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BM_CLICK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.2 Messages.BM_GETCHECK Constant

Represents the Windows message BM_GETCHECK.

Pascal

```
BM_GETCHECK = $00F0;
```

C++

```
BM_GETCHECK = 0x00F0;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BM_GETCHECK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.3 Messages.BM_GETIMAGE Constant

Represents the Windows message BM_GETIMAGE.

Pascal

```
BM_GETIMAGE = $00F6;
```

C++

```
BM_GETIMAGE = 0x00F6;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BM_GETIMAGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.4 Messages.BM_GETSTATE Constant

Represents the Windows message BM_GETSTATE.

Pascal

```
BM_GETSTATE = $00F2;
```

C++

```
BM_GETSTATE = 0x00F2;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BM_GETSTATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.5 Messages.BM_SETCHECK Constant

Represents the Windows message BM_SETCHECK.

Pascal

```
BM_SETCHECK = $00F1;
```

C++

```
BM_SETCHECK = 0x00F1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information, see the MSDN article for BM_SETCHECK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.6 Messages.BM_SETIMAGE Constant

Represents the Windows message BM_SETIMAGE.

Pascal

```
BM_SETIMAGE = $00F7;
```

C++

```
BM_SETIMAGE = 0x00F7;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BM_SETIMAGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.7 Messages.BM_SETSTATE Constant

Represents the Windows message BM_SETSTATE.

Pascal

```
BM_SETSTATE = $00F3;
```

C++

```
BM_SETSTATE = 0x00F3;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BM_SETSTATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.8 Messages.BM_SETSTYLE Constant

Represents the Windows message BM_SETSTYLE.

Pascal

```
BM_SETSTYLE = $00F4;
```

C++

```
BM_SETSTYLE = 0x00F4;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BM_SETSTYLE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.9 Messages.BN_CLICKED Constant

Represents the Windows message BN_CLICKED.

Pascal

```
BN_CLICKED = 0;
```

C++

```
BN_CLICKED = 0;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BN_CLICKED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.10 Messages.BN_DBLCLK Constant

Represents the Windows message BN_DBLCLK.

Pascal

```
BN_DBLCLK = BN_DOUBLECLICKED;
```

C++

```
BN_DBLCLK = BN_DOUBLECLICKED;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BN_DBLCLK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.11 Messages.BN_DISABLE Constant

Represents the Windows message BN_DISABLE.

Pascal

```
BN_DISABLE = 4;
```

C++

```
BN_DISABLE = 4;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BN_DISABLE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.12 Messages.BN_DOUBLECLICKED Constant

Represents the Windows message BN_DOUBLECLICKED.

Pascal

```
BN_DOUBLECLICKED = 5;
```

C++

```
BN_DOUBLECLICKED = 5;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BN_DOUBLECLICKED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.13 Messages.BN_HILITE Constant

Represents the Windows message BN_HILITE.

Pascal

```
BN_HILITE = 2;
```

C++

```
BN_HILITE = 2;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BN_HILITE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.14 Messages.BN_KILLFOCUS Constant

Represents the Windows message BN_KILLFOCUS.

Pascal

```
BN_KILLFOCUS = 7;
```

C++

```
BN_KILLFOCUS = 7;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BN_KILLFOCUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.15 Messages.BN_PAINT Constant

Represents the Windows message BN_PAINT.

Pascal

```
BN_PAINT = 1;
```

C++

```
BN_PAINT = 1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BN_PAINT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.16 Messages.BN_PUSHED Constant

Represents the Windows message BN_PUSHED.

Pascal

```
BN_PUSHED = BN_HILITE;
```

C++

```
BN_PUSHED = BN_HILITE;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BN_PUSHED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.17 Messages.BN_SETFOCUS Constant

Represents the Windows message BN_SETFOCUS.

Pascal

```
BN_SETFOCUS = 6;
```

C++

```
BN_SETFOCUS = 6;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BN_SETFOCUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.18 Messages.BN_UNHILITE Constant

Represents the Windows message BN_UNHILITE.

Pascal

```
BN_UNHILITE = 3;
```

C++

```
BN_UNHILITE = 3;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BN_UNHILITE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.19 Messages.BN_UNPUSHED Constant

Represents the Windows message BN_UNPUSHED.

Pascal

```
BN_UNPUSHED = BN_UNHILITE;
```

C++

```
BN_UNPUSHED = BN_UNHILITE;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for BN_UNPUSHED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.20 Messages.CB_ADDSTRING Constant

Represents the Windows message CB_ADDSTRING.

Pascal

```
CB_ADDSTRING = $0143;
```

C++

```
CB_ADDSTRING = 0x0143;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_ADDSTRING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.21 Messages.CB_DELETESTRING Constant

Represents the Windows message CB_DELETESTRING.

Pascal

```
CB_DELETESTRING = $0144;
```

C++

```
CB_DELETESTRING = 0x0144;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_DELETESTRING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.22 Messages.CB_DIR Constant

Represents the Windows message CB_DIR.

Pascal

```
CB_DIR = $0145;
```

C++

```
CB_DIR = 0x0145;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_DIR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.23 Messages.CB_FINDSTRING Constant

Represents the Windows message CB_FINDSTRING.

Pascal

```
CB_FINDSTRING = $014C;
```

C++

```
CB_FINDSTRING = 0x014C;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_FINDSTRING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.24 Messages.CB_FINDSTRINGEXACT Constant

Represents the Windows message CB_FINDSTRINGEXACT.

Pascal

```
CB_FINDSTRINGEXACT = $0158;
```

C++

```
CB_FINDSTRINGEXACT = 0x0158;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_FINDSTRINGEXACT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.25 Messages.CB_GETCOUNT Constant

Represents the Windows message CB_GETCOUNT.

Pascal

```
CB_GETCOUNT = $0146;
```

C++

```
CB_GETCOUNT = 0x0146;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETCOUNT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.26 Messages.CB_GETCURSEL Constant

Represents the Windows message CB_GETCURSEL.

Pascal

```
CB_GETCURSEL = $0147;
```

C++

```
CB_GETCURSEL = 0x0147;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETCURSEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.27 Messages.CB_GETDROPPEDCONTROLRECT Constant

Represents the Windows message CB_GETDROPPEDCONTROLRECT.

Pascal

```
CB_GETDROPPEDCONTROLRECT = $0152;
```

C++

```
CB_GETDROPPEDCONTROLRECT = 0x0152;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETDROPPEDCONTROLRECT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.28 Messages.CB_GETDROPPEDSTATE Constant

Represents the Windows message CB_GETDROPPEDSTATE.

Pascal

```
CB_GETDROPPEDSTATE = $0157;
```

C++

```
CB_GETDROPPEDSTATE = 0x0157;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETDROPPEDSTATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.29 Messages.CB_GETDROPPEDWIDTH Constant

Represents the Windows message CB_GETDROPPEDWIDTH.

Pascal

```
CB_GETDROPPEDWIDTH = 351;
```

C++

```
CB_GETDROPPEDWIDTH = 351;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETDROPPEDWIDTH notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.30 Messages.CB_GETEDITSEL Constant

Represents the Windows message CB_GETEDITSEL.

Pascal

```
CB_GETEDITSEL = $0140;
```

C++

```
CB_GETEDITSEL = 0x0140;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETEDITSEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.31 Messages.CB_GETEXTENDEDUI Constant

Represents the Windows message CB_GETEXTENDEDUI.

Pascal

```
CB_GETEXTENDEDUI = $0156;
```

C++

```
CB_GETEXTENDEDUI = 0x0156;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETEXTENDEDUI notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.32 Messages.CB_GETHORIZONTALEXTENT Constant

Represents the Windows message CB_GETHORIZONTALEXTENT.

Pascal

```
CB_GETHORIZONTALEXTENT = 349;
```

C++

```
CB_GETHORIZONTALEXTENT = 349;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETHORIZONTALEXTENT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.33 Messages.CB_GETITEMDATA Constant

Represents the Windows message CB_GETITEMDATA.

Pascal

```
CB_GETITEMDATA = $0150;
```

C++

```
CB_GETITEMDATA = 0x0150;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETITEMDATA notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.34 Messages.CB_GETITEMHEIGHT Constant

Represents the Windows message CB_GETITEMHEIGHT.

Pascal

```
CB_GETITEMHEIGHT = $0154;
```

C++

```
CB_GETITEMHEIGHT = 0x0154;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETITEMHEIGHT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.35 Messages.CB_GETLBTEXT Constant

Represents the Windows message CB_GETLBTEXT.

Pascal

```
CB_GETLBTEXT = $0148;
```

C++

```
CB_GETLBTEXT = 0x0148;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETLBTEXT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.36 Messages.CB_GETLBTEXTLEN Constant

Represents the Windows message CB_GETLBTEXTLEN.

Pascal

```
CB_GETLBTEXTLEN = $0149;
```

C++

```
CB_GETLBTEXTLEN = 0x0149;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETLBTEXTLEN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.37 Messages.CB_GETLOCALE Constant

Represents the Windows message CB_GETLOCALE.

Pascal

```
CB_GETLOCALE = 346;
```

C++

```
CB_GETLOCALE = 346;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETLOCALE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.38 Messages.CB_GETTOPINDEX Constant

Represents the Windows message CB_GETTOPINDEX.

Pascal

```
CB_GETTOPINDEX = 347;
```

C++

```
CB_GETTOPINDEX = 347;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_GETTOPINDEX notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.39 Messages.CB_INITSTORAGE Constant

Represents the Windows message CB_INITSTORAGE.

Pascal

```
CB_INITSTORAGE = 353;
```

C++

```
CB_INITSTORAGE = 353;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_INITSTORAGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.40 Messages.CB_INSERTSTRING Constant

Represents the Windows message CB_INSERTSTRING.

Pascal

```
CB_INSERTSTRING = $014A;
```

C++

```
CB_INSERTSTRING = 0x014A;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_INSERTSTRING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.41 Messages.CB_LIMITTEXT Constant

Represents the Windows message CB_LIMITTEXT.

Pascal

```
CB_LIMITTEXT = $0141;
```

C++

```
CB_LIMITTEXT = 0x0141;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_LIMITTEXT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.42 Messages.CB_MSGMAX Constant

Represents the Windows message CB_MSGMAX.

Pascal

```
CB_MSGMAX = 354;
```

C++

```
CB_MSGMAX = 354;
```

File

Messages

Description

This constant is used by Windows messaging. CB_MSGMAX is obsolete for Win32-Based Applications.

See Also

SendMessage (MSDN)

Handling messages

35.4.43 Messages.CB_RESETCONTENT Constant

Represents the Windows message CB_RESETCONTENT.

Pascal

```
CB_RESETCONTENT = $014B;
```

C++

```
CB_RESETCONTENT = 0x014B;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_RESETCONTENT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.44 Messages.CB_SELECTSTRING Constant

Represents the Windows message CB_SELECTSTRING.

Pascal

```
CB_SELECTSTRING = $014D;
```

C++

```
CB_SELECTSTRING = 0x014D;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_SELECTSTRING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.45 Messages.CB_SETCURSEL Constant

Represents the Windows message CB_SETCURSEL.

Pascal

```
CB_SETCURSEL = $014E;
```

C++

```
CB_SETCURSEL = 0x014E;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_SETCURSEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.46 Messages.CB_SETDROPPEDWIDTH Constant

Represents the Windows message CB_SETDROPPEDWIDTH.

Pascal

```
CB_SETDROPPEDWIDTH = 352;
```

C++

```
CB_SETDROPPEDWIDTH = 352;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_SETDROPPEDWIDTH notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.47 Messages.CB_SETEDITSEL Constant

Represents the Windows message CB_SETEDITSEL.

Pascal

```
CB_SETEDITSEL = $0142;
```

C++

```
CB_SETEDITSEL = 0x0142;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_SETEDITSEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.48 Messages.CB_SETEXTENDEDUI Constant

Represents the Windows message CB_SETEXTENDEDUI.

Pascal

```
CB_SETEXTENDEDUI = $0155;
```

C++

```
CB_SETEXTENDEDUI = 0x0155;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_SETEXTENDEDUI notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.49 Messages.CB_SETHORIZONTALEXTENT Constant

Represents the Windows message CB_SETHORIZONTALEXTENT.

Pascal

```
CB_SETHORIZONTALEXTENT = 350;
```

C++

```
CB_SETHORIZONTALEXTENT = 350;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_SETHORIZONTALEXTENT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.50 Messages.CB_SETITEMDATA Constant

Represents the Windows message CB_SETITEMDATA.

Pascal

```
CB_SETITEMDATA = $0151;
```

C++

```
CB_SETITEMDATA = 0x0151;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_SETITEMDATA notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.51 Messages.CB_SETITEMHEIGHT Constant

Represents the Windows message CB_SETITEMHEIGHT.

Pascal

```
CB_SETITEMHEIGHT = $0153;
```

C++

```
CB_SETITEMHEIGHT = 0x0153;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_SETITEMHEIGHT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.52 Messages.CB_SETLOCALE Constant

Represents the Windows message CB_SETLOCALE.

Pascal

```
CB_SETLOCALE = 345;
```

C++

```
CB_SETLOCALE = 345;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_SETLOCALE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.53 Messages.CB_SETTOPINDEX Constant

Represents the Windows message CB_SETTOPINDEX.

Pascal

```
CB_SETTOPINDEX = 348;
```

C++

```
CB_SETTOPINDEX = 348;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_SETTOPINDEX notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.54 Messages.CB_SHOWDROPDOWN Constant

Represents the Windows message CB_SHOWDROPDOWN.

Pascal

```
CB_SHOWDROPDOWN = $014F;
```

C++

```
CB_SHOWDROPDOWN = 0x014F;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CB_SHOWDROPDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.55 Messages.CBN_CLOSEUP Constant

Represents the Windows message CBN_CLOSEUP.

Pascal

```
CBN_CLOSEUP = 8;
```

C++

```
CBN_CLOSEUP = 8;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CBN_CLOSEUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.56 Messages.CBN_DBLCLK Constant

Represents the Windows message CBN_DBLCLK.

Pascal

```
CBN_DBLCLK = 2;
```

C++

```
CBN_DBLCLK = 2;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CBN_DBLCLK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.57 Messages.CBN_DROPDOWN Constant

Represents the Windows message CBN_DROPDOWN.

Pascal

```
CBN_DROPDOWN = 7;
```

C++

```
CBN_DROPDOWN = 7;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CBN_DROPDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.58 Messages.CBN_EDITCHANGE Constant

Represents the Windows message CBN_EDITCHANGE.

Pascal

```
CBN_EDITCHANGE = 5;
```

C++

```
CBN_EDITCHANGE = 5;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CBN_EDITCHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.59 Messages.CBN_EDITUPDATE Constant

Represents the Windows message CBN_EDITUPDATE.

Pascal

```
CBN_EDITUPDATE = 6;
```

C++

```
CBN_EDITUPDATE = 6;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CBN_EDITUPDATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.60 Messages.CBN_ERRSPACE Constant

Represents the Windows message CBN_ERRSPACE.

Pascal

```
CBN_ERRSPACE = (-1);
```

C++

```
CBN_ERRSPACE = (-1);
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CBN_ERRSPACE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.61 Messages.CBN_KILLFOCUS Constant

Represents the Windows message CBN_KILLFOCUS.

Pascal

```
CBN_KILLFOCUS = 4;
```

C++

```
CBN_KILLFOCUS = 4;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CBN_KILLFOCUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.62 Messages.CBN_SELCHANGE Constant

Represents the Windows message CBN_SELCHANGE.

Pascal

```
CBN_SELCHANGE = 1;
```

C++

```
CBN_SELCHANGE = 1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CBN_SELCHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.63 Messages.CBN_SELENDCANCEL Constant

Represents the Windows message CBN_SELENDCANCEL.

Pascal

```
CBN_SELENDCANCEL = 10;
```

C++

```
CBN_SELENDCANCEL = 10;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CBN_SELENDCANCEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.64 Messages.CBN_SELENDOK Constant

Represents the Windows message CBN_SELENDOK.

Pascal

```
CBN_SELENDOK = 9;
```

C++

```
CBN_SELENDOK = 9;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CBN_SELENDOK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.65 Messages.CBN_SETFOCUS Constant

Represents the Windows message CBN_SETFOCUS.

Pascal

```
CBN_SETFOCUS = 3;
```

C++

```
CBN_SETFOCUS = 3;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for CBN_SETFOCUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.66 Messages.DM_GETDEFID Constant

Represents the Windows message DM_GETDEFID.

Pascal

```
DM_GETDEFID = (WM_USER+0);
```

C++

```
DM_GETDEFID = (WM_USER+0);
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for DM_GETDEFID notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.67 Messages.DM_REPOSITION Constant

Represents the Windows message DM_REPOSITION.

Pascal

```
DM_REPOSITION = (WM_USER+2);
```

C++

```
DM_REPOSITION = (WM_USER+2);
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for DM_REPOSITION notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.68 Messages.DM_SETDEFID Constant

Represents the Windows message DM_SETDEFID.

Pascal

```
DM_SETDEFID = (WM_USER+1);
```

C++

```
DM_SETDEFID = (WM_USER+1);
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for DM_SETDEFID notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.69 Messages.EM_CANUNDO Constant

Represents the Windows message EM_CANUNDO.

Pascal

```
EM_CANUNDO = $00C6;
```

C++

```
EM_CANUNDO = 0x00C6;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_CANUNDO notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.70 Messages.EM_CHARFROMPOS Constant

Represents the Windows message EM_CHARFROMPOS.

Pascal

```
EM_CHARFROMPOS = 215;
```

C++

```
EM_CHARFROMPOS = 215;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_CHARFROMPOS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.71 Messages.EM_EMPTYUNDOBUFFER Constant

Represents the Windows message EM_EMPTYUNDOBUFFER.

Pascal

```
EM_EMPTYUNDOBUFFER = $00CD;
```

C++

```
EM_EMPTYUNDOBUFFER = 0x00CD;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_EMPTYUNDOBUFFER notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.72 Messages.EM_FMTLINES Constant

Represents the Windows message EM_FMTLINES.

Pascal

```
EM_FMTLINES = $00C8;
```

C++

```
EM_FMTLINES = 0x00C8;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_FMTLINES notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.73 Messages.EM_GETFIRSTVISIBLELINE Constant

Represents the Windows message EM_GETFIRSTVISIBLELINE.

Pascal

```
EM_GETFIRSTVISIBLELINE = $00CE;
```

C++

```
EM_GETFIRSTVISIBLELINE = 0x00CE;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETFIRSTVISIBLELINE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.74 Messages.EM_GETHANDLE Constant

Represents the Windows message EM_GETHANDLE.

Pascal

```
EM_GETHANDLE = $00BD;
```

C++

```
EM_GETHANDLE = 0x00BD;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETHANDLE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.75 Messages.EM_GETIMESTATUS Constant

Represents the Windows message EM_GETIMESTATUS.

Pascal

```
EM_GETIMESTATUS = 217;
```

C++

```
EM_GETIMESTATUS = 217;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETIMESTATUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.76 Messages.EM_GETLIMITTEXT Constant

Represents the Windows message EM_GETLIMITTEXT.

Pascal

```
EM_GETLIMITTEXT = 213;
```

C++

```
EM_GETLIMITTEXT = 213;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETLIMITTEXT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.77 Messages.EM_GETLINE Constant

Represents the Windows message EM_GETLINE.

Pascal

```
EM_GETLINE = $00C4;
```

C++

```
EM_GETLINE = 0x00C4;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETLINE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.78 Messages.EM_GETLINECOUNT Constant

Represents the Windows message EM_GETLINECOUNT.

Pascal

```
EM_GETLINECOUNT = $00BA;
```

C++

```
EM_GETLINECOUNT = 0x00BA;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETLINECOUNT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.79 Messages.EM_GETMARGINS Constant

Represents the Windows message EM_GETMARGINS.

Pascal

```
EM_GETMARGINS = 212;
```

C++

```
EM_GETMARGINS = 212;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETMARGINS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.80 Messages.EM_GETMODIFY Constant

Represents the Windows message EM_GETMODIFY.

Pascal

```
EM_GETMODIFY = $00B8;
```

C++

```
EM_GETMODIFY = 0x00B8;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETMODIFY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.81 Messages.EM_GETPASSWORDCHAR Constant

Represents the Windows message EM_GETPASSWORDCHAR.

Pascal

```
EM_GETPASSWORDCHAR = $00D2;
```

C++

```
EM_GETPASSWORDCHAR = 0x00D2;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETPASSWORDCHAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.82 Messages.EM_GETRECT Constant

Represents the Windows message EM_GETRECT.

Pascal

```
EM_GETRECT = $00B2;
```

C++

```
EM_GETRECT = 0x00B2;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETRECT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.83 Messages.EM_GETSEL Constant

Represents the Windows message EM_GETSEL.

Pascal

```
EM_GETSEL = $00B0;
```

C++

```
EM_GETSEL = 0x00B0;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETSEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.84 Messages.EM_GETTHUMB Constant

Represents the Windows message EM_GETTHUMB.

Pascal

```
EM_GETTHUMB = $00BE;
```

C++

```
EM_GETTHUMB = 0x00BE;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETTHUMB notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.85 Messages.EM_GETWORDBREAKPROC Constant

Represents the Windows message EM_GETWORDBREAKPROC.

Pascal

```
EM_GETWORDBREAKPROC = $00D1;
```

C++

```
EM_GETWORDBREAKPROC = 0x00D1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_GETWORDBREAKPROC notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.86 Messages.EM_LIMITTEXT Constant

Represents the Windows message EM_LIMITTEXT.

Pascal

```
EM_LIMITTEXT = $00C5;
```

C++

```
EM_LIMITTEXT = 0x00C5;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_LIMITTEXT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.87 Messages.EM_LINEFROMCHAR Constant

Represents the Windows message EM_LINEFROMCHAR.

Pascal

```
EM_LINEFROMCHAR = $00C9;
```

C++

```
EM_LINEFROMCHAR = 0x00C9;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_LINEFROMCHAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.88 Messages.EM_LINEINDEX Constant

Represents the Windows message EM_LINEINDEX.

Pascal

```
EM_LINEINDEX = $00BB;
```

C++

```
EM_LINEINDEX = 0x00BB;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_LINEINDEX notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.89 Messages.EM_LINELENGTH Constant

Represents the Windows message EM_LINELENGTH.

Pascal

```
EM_LINELENGTH = $00C1;
```

C++

```
EM_LINELENGTH = 0x00C1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_LINELENGTH notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.90 Messages.EM_LINESCROLL Constant

Represents the Windows message EM_LINESCROLL.

Pascal

```
EM_LINESCROLL = $00B6;
```

C++

```
EM_LINESCROLL = 0x00B6;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_LINESCROLL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.91 Messages.EM_POSFROMCHAR Constant

Represents the Windows message EM_POSFROMCHAR.

Pascal

```
EM_POSFROMCHAR = 214;
```

C++

```
EM_POSFROMCHAR = 214;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_POSFROMCHAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.92 Messages.EM_REPLACESEL Constant

Represents the Windows message EM_REPLACESEL.

Pascal

```
EM_REPLACESEL = $00C2;
```

C++

```
EM_REPLACESEL = 0x00C2;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_REPLACESEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.93 Messages.EM_SCROLL Constant

Represents the Windows message EM_SCROLL.

Pascal

```
EM_SCROLL = $00B5;
```

C++

```
EM_SCROLL = 0x00B5;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SCROLL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.94 Messages.EM_SCROLLCARET Constant

Represents the Windows message EM_SCROLLCARET.

Pascal

```
EM_SCROLLCARET = $00B7;
```

C++

```
EM_SCROLLCARET = 0x00B7;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SCROLLCARET notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.95 Messages.EM_SETHANDLE Constant

Represents the Windows message EM_SETHANDLE.

Pascal

```
EM_SETHANDLE = $00BC;
```

C++

```
EM_SETHANDLE = 0x00BC;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETHANDLE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.96 Messages.EM_SETIMESTATUS Constant

Represents the Windows message EM_SETIMESTATUS.

Pascal

```
EM_SETIMESTATUS = 216;
```

C++

```
EM_SETIMESTATUS = 216;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETIMESTATUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.97 Messages.EM_SETLIMITTEXT Constant

Represents the Windows message EM_SETLIMITTEXT.

Pascal

```
EM_SETLIMITTEXT = EM_LIMITTEXT;
```

C++

```
EM_SETLIMITTEXT = EM_LIMITTEXT;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETLIMITTEXT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.98 Messages.EM_SETMARGINS Constant

Represents the Windows message EM_SETMARGINS.

Pascal

```
EM_SETMARGINS = 211;
```

C++

```
EM_SETMARGINS = 211;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETMARGINS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.99 Messages.EM_SETMODIFY Constant

Represents the Windows message EM_SETMODIFY.

Pascal

```
EM_SETMODIFY = $00B9;
```

C++

```
EM_SETMODIFY = 0x00B9;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETMODIFY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.100 Messages.EM_SETPASSWORDCHAR Constant

Represents the Windows message EM_SETPASSWORDCHAR.

Pascal

```
EM_SETPASSWORDCHAR = $00CC;
```

C++

```
EM_SETPASSWORDCHAR = 0x00CC;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETPASSWORDCHAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.101 Messages.EM_SETREADONLY Constant

Represents the Windows message EM_SETREADONLY.

Pascal

```
EM_SETREADONLY = $00CF;
```

C++

```
EM_SETREADONLY = 0x00CF;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETREADONLY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.102 Messages.EM_SETRECT Constant

Represents the Windows message EM_SETRECT.

Pascal

```
EM_SETRECT = $00B3;
```

C++

```
EM_SETRECT = 0x00B3;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETRECT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.103 Messages.EM_SETRECTNP Constant

Represents the Windows message EM_SETRECTNP.

Pascal

```
EM_SETRECTNP = $00B4;
```

C++

```
EM_SETRECTNP = 0x00B4;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETRECTNP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.104 Messages.EM_SETSEL Constant

Represents the Windows message EM_SETSEL.

Pascal

```
EM_SETSEL = $00B1;
```

C++

```
EM_SETSEL = 0x00B1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETSEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.105 Messages.EM_SETTABSTOPS Constant

Represents the Windows message EM_SETTABSTOPS.

Pascal

```
EM_SETTABSTOPS = $00CB;
```

C++

```
EM_SETTABSTOPS = 0x00CB;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETTABSTOPS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.106 Messages.EM_SETWORDBREAKPROC Constant

Represents the Windows message EM_SETWORDBREAKPROC.

Pascal

```
EM_SETWORDBREAKPROC = $00D0;
```

C++

```
EM_SETWORDBREAKPROC = 0x00D0;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_SETWORDBREAKPROC notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.107 Messages.EM_UNDO Constant

Represents the Windows message EM_UNDO.

Pascal

```
EM_UNDO = $00C7;
```

C++

```
EM_UNDO = 0x00C7;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EM_UNDO notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.108 Messages.EN_CHANGE Constant

Represents the Windows message EN_CHANGE.

Pascal

```
EN_CHANGE = $0300;
```

C++

```
EN_CHANGE = 0x0300;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EN_CHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.109 Messages.EN_ERRSPACE Constant

Represents the Windows message EN_ERRSPACE.

Pascal

```
EN_ERRSPACE = $0500;
```

C++

```
EN_ERRSPACE = 0x0500;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EN_ERRSPACE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.110 Messages.EN_HSCROLL Constant

Represents the Windows message EN_HSCROLL.

Pascal

```
EN_HSCROLL = $0601;
```

C++

```
EN_HSCROLL = 0x0601;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EN_HSCROLL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.111 Messages.EN_KILLFOCUS Constant

Represents the Windows message EN_KILLFOCUS.

Pascal

```
EN_KILLFOCUS = $0200;
```

C++

```
EN_KILLFOCUS = 0x0200;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EN_KILLFOCUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.112 Messages.EN_MAXTEXT Constant

Represents the Windows message EN_MAXTEXT.

Pascal

```
EN_MAXTEXT = $0501;
```

C++

```
EN_MAXTEXT = 0x0501;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EN_MAXTEXT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.113 Messages.EN_SETFOCUS Constant

Represents the Windows message EN_SETFOCUS.

Pascal

```
EN_SETFOCUS = $0100;
```

C++

```
EN_SETFOCUS = 0x0100;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EN_SETFOCUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.114 Messages.EN_UPDATE Constant

Represents the Windows message EN_UPDATE.

Pascal

```
EN_UPDATE = $0400;
```

C++

```
EN_UPDATE = 0x0400;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EN_UPDATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.115 Messages.EN_VSCROLL Constant

Represents the Windows message EN_VSCROLL.

Pascal

```
EN_VSCROLL = $0602;
```

C++

```
EN_VSCROLL = 0x0602;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for EN_VSCROLL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.116 Messages.LB_ADDFILE Constant

Represents the Windows message LB_ADDFILE.

Pascal

```
LB_ADDFILE = $0196;
```

C++

```
LB_ADDFILE = 0x0196;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_ADDFILE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.117 Messages.LB_ADDSTRING Constant

Represents the Windows message LB_ADDSTRING.

Pascal

```
LB_ADDSTRING = $0180;
```

C++

```
LB_ADDSTRING = 0x0180;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_ADDSTRING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.118 Messages.LB_DELETESTRING Constant

Represents the Windows message LB_DELETESTRING.

Pascal

```
LB_DELETESTRING = $0182;
```

C++

```
LB_DELETESTRING = 0x0182;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_DELETESTRING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.119 Messages.LB_DIR Constant

Represents the Windows message LB_DIR.

Pascal

```
LB_DIR = $018D;
```

C++

```
LB_DIR = 0x018D;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_DIR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.120 Messages.LB_FINDSTRING Constant

Represents the Windows message LB_FINDSTRING.

Pascal

```
LB_FINDSTRING = $018F;
```

C++

```
LB_FINDSTRING = 0x018F;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_FINDSTRING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.121 Messages.LB_FINDSTRINGEXACT Constant

Represents the Windows message LB_FINDSTRINGEXACT.

Pascal

```
LB_FINDSTRINGEXACT = $01A2;
```

C++

```
LB_FINDSTRINGEXACT = 0x01A2;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_FINDSTRINGEXACT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.122 Messages.LB_GETANCHORINDEX Constant

Represents the Windows message LB_GETANCHORINDEX.

Pascal

```
LB_GETANCHORINDEX = $019D;
```

C++

```
LB_GETANCHORINDEX = 0x019D;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETANCHORINDEX notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.123 Messages.LB_GETCARETINDEX Constant

Represents the Windows message LB_GETCARETINDEX.

Pascal

```
LB_GETCARETINDEX = $019F;
```

C++

```
LB_GETCARETINDEX = 0x019F;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETCARETINDEX notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.124 Messages.LB_GETCOUNT Constant

Represents the Windows message LB_GETCOUNT.

Pascal

```
LB_GETCOUNT = $018B;
```

C++

```
LB_GETCOUNT = 0x018B;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETCOUNT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.125 Messages.LB_GETCURSEL Constant

Represents the Windows message LB_GETCURSEL.

Pascal

```
LB_GETCURSEL = $0188;
```

C++

```
LB_GETCURSEL = 0x0188;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETCURSEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.126 Messages.LB_GETHORIZONTALEXTENT Constant

Represents the Windows message LB_GETHORIZONTALEXTENT.

Pascal

```
LB_GETHORIZONTALEXTENT = $0193;
```

C++

```
LB_GETHORIZONTALEXTENT = 0x0193;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETHORIZONTALEXTENT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.127 Messages.LB_GETITEMDATA Constant

Represents the Windows message LB_GETITEMDATA.

Pascal

```
LB_GETITEMDATA = $0199;
```

C++

```
LB_GETITEMDATA = 0x0199;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETITEMDATA notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.128 Messages.LB_GETITEMHEIGHT Constant

Represents the Windows message LB_GETITEMHEIGHT.

Pascal

```
LB_GETITEMHEIGHT = $01A1;
```

C++

```
LB_GETITEMHEIGHT = 0x01A1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETITEMHEIGHT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.129 Messages.LB_GETITEMRECT Constant

Represents the Windows message LB_GETITEMRECT.

Pascal

```
LB_GETITEMRECT = $0198;
```

C++

```
LB_GETITEMRECT = 0x0198;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETITEMRECT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.130 Messages.LB_GETLOCALE Constant

Represents the Windows message LB_GETLOCALE.

Pascal

```
LB_GETLOCALE = $01A6;
```

C++

```
LB_GETLOCALE = 0x01A6;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETLOCALE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.131 Messages.LB_GETSEL Constant

Represents the Windows message LB_GETSEL.

Pascal

```
LB_GETSEL = $0187;
```

C++

```
LB_GETSEL = 0x0187;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETSEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.132 Messages.LB_GETSELCOUNT Constant

Represents the Windows message LB_GETSELCOUNT.

Pascal

```
LB_GETSELCOUNT = $0190;
```

C++

```
LB_GETSELCOUNT = 0x0190;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETSELCOUNT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.133 Messages.LB_GETSELITEMS Constant

Represents the Windows message LB_GETSELITEMS.

Pascal

```
LB_GETSELITEMS = $0191;
```

C++

```
LB_GETSELITEMS = 0x0191;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETSELITEMS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.134 Messages.LB_GETTEXT Constant

Represents the Windows message LB_GETTEXT.

Pascal

```
LB_GETTEXT = $0189;
```

C++

```
LB_GETTEXT = 0x0189;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETTEXT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.135 Messages.LB_GETTEXTLEN Constant

Represents the Windows message LB_GETTEXTLEN.

Pascal

```
LB_GETTEXTLEN = $018A;
```

C++

```
LB_GETTEXTLEN = 0x018A;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETTEXTLEN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.136 Messages.LB_GETTOPINDEX Constant

Represents the Windows message LB_GETTOPINDEX.

Pascal

```
LB_GETTOPINDEX = $018E;
```

C++

```
LB_GETTOPINDEX = 0x018E;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_GETTOPINDEX notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.137 Messages.LB_INITSTORAGE Constant

Represents the Windows message LB_INITSTORAGE.

Pascal

```
LB_INITSTORAGE = $01A8;
```

C++

```
LB_INITSTORAGE = 0x01A8;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_INITSTORAGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.138 Messages.LB_INSERTSTRING Constant

Represents the Windows message LB_INSERTSTRING.

Pascal

```
LB_INSERTSTRING = $0181;
```

C++

```
LB_INSERTSTRING = 0x0181;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_INSERTSTRING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.139 Messages.LB_ITEMFROMPOINT Constant

Represents the Windows message LB_ITEMFROMPOINT.

Pascal

```
LB_ITEMFROMPOINT = $01A9;
```

C++

```
LB_ITEMFROMPOINT = 0x01A9;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_ITEMFROMPOINT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.140 Messages.LB_MSGMAX Constant

Represents the Windows message LB_MSGMAX.

Pascal

```
LB_MSGMAX = 432;
```

C++

```
LB_MSGMAX = 432;
```

File

Messages

Description

This constant is used by Windows messaging. LB_MSGMAX is obsolete for Win32-Based Applications.

See Also

SendMessage (MSDN)

Handling messages

35.4.141 Messages.LB_RESETCONTENT Constant

Represents the Windows message LB_RESETCONTENT.

Pascal

```
LB_RESETCONTENT = $0184;
```

C++

```
LB_RESETCONTENT = 0x0184;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_RESETCONTENT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.142 Messages.LB_SELECTSTRING Constant

Represents the Windows message LB_SELECTSTRING.

Pascal

```
LB_SELECTSTRING = $018C;
```

C++

```
LB_SELECTSTRING = 0x018C;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SELECTSTRING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.143 Messages.LB_SELITEMRANGE Constant

Represents the Windows message LB_SELITEMRANGE.

Pascal

```
LB_SELITEMRANGE = $019B;
```

C++

```
LB_SELITEMRANGE = 0x019B;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SELITEMRANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.144 Messages.LB_SELITEMRANGEEX Constant

Represents the Windows message LB_SELITEMRANGEEX.

Pascal

```
LB_SELITEMRANGEEX = $0183;
```

C++

```
LB_SELITEMRANGEEX = 0x0183;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SELITEMRANGEEX notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.145 Messages.LB_SETANCHORINDEX Constant

Represents the Windows message LB_SETANCHORINDEX.

Pascal

```
LB_SETANCHORINDEX = $019C;
```

C++

```
LB_SETANCHORINDEX = 0x019C;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETANCHORINDEX notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.146 Messages.LB_SETCARETINDEX Constant

Represents the Windows message LB_SETCARETINDEX.

Pascal

```
LB_SETCARETINDEX = $019E;
```

C++

```
LB_SETCARETINDEX = 0x019E;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETCARETINDEX notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.147 Messages.LB_SETCOLUMNWIDTH Constant

Represents the Windows message LB_SETCOLUMNWIDTH.

Pascal

```
LB_SETCOLUMNWIDTH = $0195;
```

C++

```
LB_SETCOLUMNWIDTH = 0x0195;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETCOLUMNWIDTH notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.148 Messages.LB_SETCOUNT Constant

Represents the Windows message LB_SETCOUNT.

Pascal

```
LB_SETCOUNT = $01A7;
```

C++

```
LB_SETCOUNT = 0x01A7;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETCOUNT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.149 Messages.LB_SETCURSEL Constant

Represents the Windows message LB_SETCURSEL.

Pascal

```
LB_SETCURSEL = $0186;
```

C++

```
LB_SETCURSEL = 0x0186;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETCURSEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.150 Messages.LB_SETHORIZONTALEXTENT Constant

Represents the Windows message LB_SETHORIZONTALEXTENT.

Pascal

```
LB_SETHORIZONTALEXTENT = $0194;
```

C++

```
LB_SETHORIZONTALEXTENT = 0x0194;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETHORIZONTALEXTENT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.151 Messages.LB_SETITEMDATA Constant

Represents the Windows message LB_SETITEMDATA.

Pascal

```
LB_SETITEMDATA = $019A;
```

C++

```
LB_SETITEMDATA = 0x019A;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETITEMDATA notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.152 Messages.LB_SETITEMHEIGHT Constant

Represents the Windows message LB_SETITEMHEIGHT.

Pascal

```
LB_SETITEMHEIGHT = $01A0;
```

C++

```
LB_SETITEMHEIGHT = 0x01A0;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETITEMHEIGHT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.153 Messages.LB_SETLOCALE Constant

Represents the Windows message LB_SETLOCALE.

Pascal

```
LB_SETLOCALE = $01A5;
```

C++

```
LB_SETLOCALE = 0x01A5;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETLOCALE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.154 Messages.LB_SETSEL Constant

Represents the Windows message LB_SETSEL.

Pascal

```
LB_SETSEL = $0185;
```

C++

```
LB_SETSEL = 0x0185;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETSEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.155 Messages.LB_SETTABSTOPS Constant

Represents the Windows message LB_SETTABSTOPS.

Pascal

```
LB_SETTABSTOPS = $0192;
```

C++

```
LB_SETTABSTOPS = 0x0192;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETTABSTOPS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.156 Messages.LB_SETTOPINDEX Constant

Represents the Windows message LB_SETTOPINDEX.

Pascal

```
LB_SETTOPINDEX = $0197;
```

C++

```
LB_SETTOPINDEX = 0x0197;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LB_SETTOPINDEX notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.157 Messages.LBN_DBLCLK Constant

Represents the Windows message LBN_DBLCLK.

Pascal

```
LBN_DBLCLK = 2;
```

C++

```
LBN_DBLCLK = 2;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LBN_DBLCLK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.158 Messages.LBN_ERRSPACE Constant

Represents the Windows message LBN_ERRSPACE.

Pascal

```
LBN_ERRSPACE = (-2);
```

C++

```
LBN_ERRSPACE = (-2);
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LBN_ERRSPACE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.159 Messages.LBN_KILLFOCUS Constant

Represents the Windows message LBN_KILLFOCUS.

Pascal

```
LBN_KILLFOCUS = 5;
```

C++

```
LBN_KILLFOCUS = 5;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LBN_KILLFOCUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.160 Messages.LBN_SELCANCEL Constant

Represents the Windows message LBN_SELCANCEL.

Pascal

```
LBN_SELCANCEL = 3;
```

C++

```
LBN_SELCANCEL = 3;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LBN_SELCANCEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.161 Messages.LBN_SELCHANGE Constant

Represents the Windows message LBN_SELCHANGE.

Pascal

```
LBN_SELCHANGE = 1;
```

C++

```
LBN_SELCHANGE = 1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LBN_SELCHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.162 Messages.LBN_SETFOCUS Constant

Represents the Windows message LBN_SETFOCUS.

Pascal

```
LBN_SETFOCUS = 4;
```

C++

```
LBN_SETFOCUS = 4;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for LBN_SETFOCUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.163 Messages.PSM_PAGEINFO Constant

Represents the Windows message PSM_PAGEINFO.

Pascal

```
PSM_PAGEINFO = (WM_USER+100);
```

C++

```
PSM_PAGEINFO = (WM_USER+100);
```

File

Messages

Description

This constant is used by Windows messaging. PSM_PAGEINFO is obsolete for Win32-Based Applications.

See Also

SendMessage (MSDN)

Handling messages

35.4.164 Messages.PSM_SHEETINFO Constant

Represents the Windows message PSM_SHEETINFO.

Pascal

```
PSM_SHEETINFO = (WM_USER+101);
```

C++

```
PSM_SHEETINFO = (WM_USER+101);
```

File

Messages

Description

This constant is used by Windows messaging. PSM_SHEETINFO is obsolete for Win32-Based Applications.

See Also

SendMessage (MSDN)

Handling messages

35.4.165 Messages.SBM_ENABLE_ARROWS Constant

Represents the Windows message SBM_ENABLE_ARROWS.

Pascal

```
SBM_ENABLE_ARROWS = 228;
```

C++

```
SBM_ENABLE_ARROWS = 228;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for SBM_ENABLE_ARROWS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.166 Messages.SBM_GETPOS Constant

Represents the Windows message SBM_GETPOS.

Pascal

```
SBM_GETPOS = 225;
```

C++

```
SBM_GETPOS = 225;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for SBM_GETPOS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.167 Messages.SBM_GETRANGE Constant

Represents the Windows message SBM_GETRANGE.

Pascal

```
SBM_GETRANGE = 227;
```

C++

```
SBM_GETRANGE = 227;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for SBM_GETRANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.168 Messages.SBM_GETSCROLLBARINFO Constant

Represents the Windows message SBM_GETSCROLLBARINFO.

Pascal

```
SBM_GETSCROLLBARINFO = 235;
```

C++

```
SBM_GETSCROLLBARINFO = 235;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for SBM_GETSCROLLBARINFO notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.169 Messages.SBM_GETSCROLLINFO Constant

Represents the Windows message SBM_GETSCROLLINFO.

Pascal

```
SBM_GETSCROLLINFO = 234;
```

C++

```
SBM_GETSCROLLINFO = 234;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for SBM_GETSCROLLINFO notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.170 Messages.SBM_SETPOS Constant

Represents the Windows message SBM_SETPOS.

Pascal

```
SBM_SETPOS = 224;
```

C++

```
SBM_SETPOS = 224;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for SBM_SETPOS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.171 Messages.SBM_SETRANGE Constant

Represents the Windows message SBM_SETRANGE.

Pascal

```
SBM_SETRANGE = 226;
```

C++

```
SBM_SETRANGE = 226;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for SBM_SETRANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.172 Messages.SBM_SETRANGEREDRAW Constant

Represents the Windows message SBM_SETRANGEREDRAW.

Pascal

```
SBM_SETRANGEREDRAW = 230;
```

C++

```
SBM_SETRANGEREDRAW = 230;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for SBM_SETRANGEREDRAW notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.173 Messages.SBM_SETSCROLLINFO Constant

Represents the Windows message SBM_SETSCROLLINFO.

Pascal

```
SBM_SETSCROLLINFO = 233;
```

C++

```
SBM_SETSCROLLINFO = 233;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for SBM_SETSCROLLINFO notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.174 Messages.WM_ACTIVATE Constant

Represents the Windows message WM_ACTIVATE.

Pascal

```
WM_ACTIVATE = $0006;
```

C++

```
WM_ACTIVATE = 0x0006;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_ACTIVATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.175 Messages.WM_ACTIVATEAPP Constant

Represents the Windows message WM_ACTIVATEAPP.

Pascal

```
WM_ACTIVATEAPP = $001C;
```

C++

```
WM_ACTIVATEAPP = 0x001C;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_ACTIVATEAPP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.176 Messages.WM_APP Constant

Represents the Windows message WM_APP.

Pascal

```
WM_APP = $8000;
```

C++

```
WM_APP = 0x8000;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_APP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.177 Messages.WM_APPCOMMAND Constant

Represents the Windows message WM_APPCOMMAND.

Pascal

```
WM_APPCOMMAND = $0319;
```

C++

```
WM_APPCOMMAND = 0x0319;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_APPCOMMAND notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.178 Messages.WM_ASKCBFORMATNAME Constant

Represents the Windows message WM_ASKCBFORMATNAME.

Pascal

```
WM_ASKCBFORMATNAME = $030C;
```

C++

```
WM_ASKCBFORMATNAME = 0x030C;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_ASKCBFORMATNAME notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.179 Messages.WM_CANCELJOURNAL Constant

Represents the Windows message WM_CANCELJOURNAL.

Pascal

```
WM_CANCELJOURNAL = $004B;
```

C++

```
WM_CANCELJOURNAL = 0x004B;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CANCELJOURNAL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.180 Messages.WM_CANCELMODE Constant

Represents the Windows message WM_CANCELMODE.

Pascal

```
WM_CANCELMODE = $001F;
```

C++

```
WM_CANCELMODE = 0x001F;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CANCELMODE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.181 Messages.WM_CAPTURECHANGED Constant

Represents the Windows message WM_CAPTURECHANGED.

Pascal

```
WM_CAPTURECHANGED = 533;
```

C++

```
WM_CAPTURECHANGED = 533;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CAPTURECHANGED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.182 Messages.WM_CHANGECBCHAIN Constant

Represents the Windows message WM_CHANGECBCHAIN.

Pascal

```
WM_CHANGECBCHAIN = $030D;
```

C++

```
WM_CHANGECBCHAIN = 0x030D;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CHANGECBCHAIN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.183 Messages.WM_CHANGEUISTATE Constant

Represents the Windows message WM_CHANGEUISTATE.

Pascal

```
WM_CHANGEUISTATE = $0127;
```

C++

```
WM_CHANGEUISTATE = 0x0127;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CHANGEUISTATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.184 Messages.WM_CHAR Constant

Represents the Windows message WM_CHAR.

Pascal

```
WM_CHAR = $0102;
```

C++

```
WM_CHAR = 0x0102;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CHAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.185 Messages.WM_CHARTOITEM Constant

Represents the Windows message WM_CHARTOITEM.

Pascal

```
WM_CHARTOITEM = $002F;
```

C++

```
WM_CHARTOITEM = 0x002F;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CHARTOITEM notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.186 Messages.WM_CHILDACTIVATE Constant

Represents the Windows message WM_CHILDACTIVATE.

Pascal

```
WM_CHILDACTIVATE = $0022;
```

C++

```
WM_CHILDACTIVATE = 0x0022;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CHILDACTIVATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.187 Messages.WM_CLEAR Constant

Represents the Windows message WM_CLEAR.

Pascal

```
WM_CLEAR = $0303;
```

C++

```
WM_CLEAR = 0x0303;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CLEAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.188 Messages.WM_CLOSE Constant

Represents the Windows message WM_CLOSE.

Pascal

```
WM_CLOSE = $0010;
```

C++

```
WM_CLOSE = 0x0010;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CLOSE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.189 Messages.WM_COALESCE_FIRST Constant

Represents the Windows message WM_COALESCE_FIRST.

Pascal

```
WM_COALESCE_FIRST = $0390;
```

C++

```
WM_COALESCE_FIRST = 0x0390;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.190 Messages.WM_COALESCE_LAST Constant

Represents the Windows message WM_COALESCE_LAST.

Pascal

```
WM_COALESCE_LAST = $039F;
```

C++

```
WM_COALESCE_LAST = 0x039F;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.191 Messages.WM_COMMAND Constant

Represents the Windows message WM_COMMAND.

Pascal

```
WM_COMMAND = $0111;
```

C++

```
WM_COMMAND = 0x0111;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_COMMAND notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.192 Messages.WM_COMMNOTIFY Constant

Represents the Windows message WM_COMMNOTIFY.

Pascal

```
WM_COMMNOTIFY = $0044;
```

C++

```
WM_COMMNOTIFY = 0x0044;
```

File

Messages

Description

This constant is used by Windows messaging. WM_COMMNOTIFY is obsolete for Win32-Based Applications.

See Also

SendMessage (MSDN)

Handling messages

35.4.193 Messages.WM_COMPACTING Constant

Represents the Windows message WM_COMPACTING.

Pascal

```
WM_COMPACTING = $0041;
```

C++

```
WM_COMPACTING = 0x0041;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_COMPACTING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.194 Messages.WM_COMPAREITEM Constant

Represents the Windows message WM_COMPAREITEM.

Pascal

```
WM_COMPAREITEM = $0039;
```

C++

```
WM_COMPAREITEM = 0x0039;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_COMPAREITEM notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.195 Messages.WM_CONTEXTMENU Constant

Represents the Windows message WM_CONTEXTMENU.

Pascal

```
WM_CONTEXTMENU = $007B;
```

C++

```
WM_CONTEXTMENU = 0x007B;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CONTEXTMENU notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.196 Messages.WM_COPY Constant

Represents the Windows message WM_COPY.

Pascal

```
WM_COPY = $0301;
```

C++

```
WM_COPY = 0x0301;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_COPY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.197 Messages.WM_COPYDATA Constant

Represents the Windows message WM_COPYDATA.

Pascal

```
WM_COPYDATA = $004A;
```

C++

```
WM_COPYDATA = 0x004A;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_COPYDATA notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.198 Messages.WM_CREATE Constant

Represents the Windows message WM_CREATE.

Pascal

```
WM_CREATE = $0001;
```

C++

```
WM_CREATE = 0x0001;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CREATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.199 Messages.WM_CTLCOLOR Constant

Represents the Windows message WM_CTLCOLOR.

Pascal

```
WM_CTLCOLOR = $0019;
```

C++

```
WM_CTLCOLOR = 0x0019;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CTLCOLOR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.200 Messages.WM_CTLCOLORBTN Constant

Represents the Windows message WM_CTLCOLORBTN.

Pascal

```
WM_CTLCOLORBTN = $0135;
```

C++

```
WM_CTLCOLORBTN = 0x0135;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CTLCOLORBTN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.201 Messages.WM_CTLCOLORDLG Constant

Represents the Windows message WM_CTLCOLORDLG.

Pascal

```
WM_CTLCOLORDLG = $0136;
```

C++

```
WM_CTLCOLORDLG = 0x0136;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CTLCOLORDLG notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.202 Messages.WM_CTLCOLOREDIT Constant

Represents the Windows message WM_CTLCOLOREDIT.

Pascal

```
WM_CTLCOLOREDIT = $0133;
```

C++

```
WM_CTLCOLOREDIT = 0x0133;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CTLCOLOREDIT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.203 Messages.WM_CTLCOLORLISTBOX Constant

Represents the Windows message WM_CTLCOLORLISTBOX.

Pascal

```
WM_CTLCOLORLISTBOX = $0134;
```

C++

```
WM_CTLCOLORLISTBOX = 0x0134;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CTLCOLORLISTBOX

notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.204 Messages.WM_CTLCOLORMSGBOX Constant

Represents the Windows message WM_CTLCOLORMSGBOX.

Pascal

```
WM_CTLCOLORMSGBOX = $0132;
```

C++

```
WM_CTLCOLORMSGBOX = 0x0132;
```

File

Messages

Description

This constant is used by Windows messaging. WM_CTLCOLORMSGBOX is obsolete for Win32-Based Applications.

See Also

SendMessage (MSDN)

Handling messages

35.4.205 Messages.WM_CTLCOLORSCROLLBAR Constant

Represents the Windows message WM_CTLCOLORSCROLLBAR.

Pascal

```
WM_CTLCOLORSCROLLBAR = $0137;
```

C++

```
WM_CTLCOLORSCROLLBAR = 0x0137;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CTLCOLORSCROLLBAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.206 Messages.WM_CTLCOLORSTATIC Constant

Represents the Windows message WM_CTLCOLORSTATIC.

Pascal

```
WM_CTLCOLORSTATIC = $0138;
```

C++

```
WM_CTLCOLORSTATIC = 0x0138;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CTLCOLORSTATIC notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.207 Messages.WM_CUT Constant

Represents the Windows message WM_CUT.

Pascal

```
WM_CUT = $0300;
```

C++

```
WM_CUT = 0x0300;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_CUT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.208 Messages.WM_DDE_ACK Constant

Represents the Windows message WM_DDE_ACK.

Pascal

```
WM_DDE_ACK = WM_DDE_FIRST + 4;
```

C++

```
WM_DDE_ACK = WM_DDE_FIRST + 4;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DDE_ACK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.209 Messages.WM_DDE_ADVISE Constant

Represents the Windows message WM_DDE_ADVISE.

Pascal

```
WM_DDE_ADVISE = WM_DDE_FIRST + 2;
```

C++

```
WM_DDE_ADVISE = WM_DDE_FIRST + 2;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DDE_ADVISE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.210 Messages.WM_DDE_DATA Constant

Represents the Windows message WM_DDE_DATA.

Pascal

```
WM_DDE_DATA = WM_DDE_FIRST + 5;
```

C++

```
WM_DDE_DATA = WM_DDE_FIRST + 5;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DDE_DATA notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.211 Messages.WM_DDE_EXECUTE Constant

Represents the Windows message WM_DDE_EXECUTE.

Pascal

```
WM_DDE_EXECUTE = WM_DDE_FIRST + 8;
```

C++

```
WM_DDE_EXECUTE = WM_DDE_FIRST + 8;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DDE_EXECUTE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.212 Messages.WM_DDE_FIRST Constant

Represents the Windows message WM_DDE_FIRST.

Pascal

```
WM_DDE_FIRST = $03E0;
```

C++

```
WM_DDE_FIRST = 0x03E0;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.213 Messages.WM_DDE_INITIATE Constant

Represents the Windows message WM_DDE_INITIATE.

Pascal

```
WM_DDE_INITIATE = WM_DDE_FIRST + 0;
```

C++

```
WM_DDE_INITIATE = WM_DDE_FIRST + 0;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DDE_INITIATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.214 Messages.WM_DDE_LAST Constant

Represents the Windows message WM_DDE_LAST.

Pascal

```
WM_DDE_LAST = WM_DDE_FIRST + 8;
```

C++

```
WM_DDE_LAST = WM_DDE_FIRST + 8;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.215 Messages.WM_DDE_POKE Constant

Represents the Windows message WM_DDE_POKE.

Pascal

```
WM_DDE_POKE = WM_DDE_FIRST + 7;
```

C++

```
WM_DDE_POKE = WM_DDE_FIRST + 7;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DDE_POKE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.216 Messages.WM_DDE_REQUEST Constant

Represents the Windows message WM_DDE_REQUEST.

Pascal

```
WM_DDE_REQUEST = WM_DDE_FIRST + 6;
```

C++

```
WM_DDE_REQUEST = WM_DDE_FIRST + 6;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DDE_REQUEST notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.217 Messages.WM_DDE_TERMINATE Constant

Represents the Windows message WM_DDE_TERMINATE.

Pascal

```
WM_DDE_TERMINATE = WM_DDE_FIRST + 1;
```

C++

```
WM_DDE_TERMINATE = WM_DDE_FIRST + 1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DDE_TERMINATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.218 Messages.WM_DDE_UNADVISE Constant

Represents the Windows message WM_DDE_UNADVISE.

Pascal

```
WM_DDE_UNADVISE = WM_DDE_FIRST + 3;
```

C++

```
WM_DDE_UNADVISE = WM_DDE_FIRST + 3;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DDE_UNADVISE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.219 Messages.WM_DEADCHAR Constant

Represents the Windows message WM_DEADCHAR.

Pascal

```
WM_DEADCHAR = $0103;
```

C++

```
WM_DEADCHAR = 0x0103;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DEADCHAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.220 Messages.WM_DELETEITEM Constant

Represents the Windows message WM_DELETEITEM.

Pascal

```
WM_DELETEITEM = $002D;
```

C++

```
WM_DELETEITEM = 0x002D;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DELETEITEM notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.221 Messages.WM_DESTROY Constant

Represents the Windows message WM_DESTROY.

Pascal

```
WM_DESTROY = $0002;
```

C++

```
WM_DESTROY = 0x0002;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DESTROY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.222 Messages.WM_DESTROYCLIPBOARD Constant

Represents the Windows message WM_DESTROYCLIPBOARD.

Pascal

```
WM_DESTROYCLIPBOARD = $0307;
```

C++

```
WM_DESTROYCLIPBOARD = 0x0307;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DESTROYCLIPBOARD notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.223 Messages.WM_DEVICECHANGE Constant

Represents the Windows message WM_DEVICECHANGE.

Pascal

```
WM_DEVICECHANGE = 537;
```

C++

```
WM_DEVICECHANGE = 537;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DEVICECHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.224 Messages.WM_DEVMODECHANGE Constant

Represents the Windows message WM_DEVMODECHANGE.

Pascal

```
WM_DEVMODECHANGE = $001B;
```

C++

```
WM_DEVMODECHANGE = 0x001B;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DEVMODECHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.225 Messages.WM_DISPLAYCHANGE Constant

Represents the Windows message WM_DISPLAYCHANGE.

Pascal

```
WM_DISPLAYCHANGE = $007E;
```

C++

```
WM_DISPLAYCHANGE = 0x007E;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DISPLAYCHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.226 Messages.WM_DRAWCLIPBOARD Constant

Represents the Windows message WM_DRAWCLIPBOARD.

Pascal

```
WM_DRAWCLIPBOARD = $0308;
```

C++

```
WM_DRAWCLIPBOARD = 0x0308;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DRAWCLIPBOARD notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.227 Messages.WM_DRAWITEM Constant

Represents the Windows message WM_DRAWITEM.

Pascal

```
WM_DRAWITEM = $002B;
```

C++

```
WM_DRAWITEM = 0x002B;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DRAWITEM notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.228 Messages.WM_DROPFILES Constant

Represents the Windows message WM_DROPFILES.

Pascal

```
WM_DROPFILES = $0233;
```

C++

```
WM_DROPFILES = 0x0233;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DROPFILES notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.229 Messages.WM_DWMCOLORIZATIONCOLORCHANGED Constant

Represents the Windows message WM_DWMCOLORIZATIONCOLORCHANGED.

Pascal

```
WM_DWMCOLORIZATIONCOLORCHANGED = $0320;
```

C++

```
WM_DWMCOLORIZATIONCOLORCHANGED = 0x0320;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DWMCOLORIZATIONCOLORCHANGED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.230 Messages.WM_DWMCOMPOSITIONCHANGED Constant

Represents the Windows message WM_DWMCOMPOSITIONCHANGED.

Pascal

```
WM_DWMCOMPOSITIONCHANGED = $031E;
```

C++

```
WM_DWMCOMPOSITIONCHANGED = 0x031E;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DWMCOMPOSITIONCHANGED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.231 Messages.WM_DWMNCRENDERINGCHANGED Constant

Represents the Windows message WM_DWMNCRENDERINGCHANGED.

Pascal

```
WM_DWMNCRENDERINGCHANGED = $031F;
```

C++

```
WM_DWMNCRENDERINGCHANGED = 0x031F;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DWMNCRENDERINGCHANGED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.232 Messages.WM_DWMWINDOWMAXIMIZEDCHANGE Constant

Represents the Windows message WM_DWMWINDOWMAXIMIZEDCHANGE.

Pascal

```
WM_DWMWINDOWMAXIMIZEDCHANGE = $0321;
```

C++

```
WM_DWMWINDOWMAXIMIZEDCHANGE = 0x0321;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_DWMWINDOWMAXIMIZEDCHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.233 Messages.WM_ENABLE Constant

Represents the Windows message WM_ENABLE.

Pascal

```
WM_ENABLE = $000A;
```

C++

```
WM_ENABLE = 0x000A;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_ENABLE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.234 Messages.WM_ENDSESSION Constant

Represents the Windows message WM_ENDSESSION.

Pascal

```
WM_ENDSESSION = $0016;
```

C++

```
WM_ENDSESSION = 0x0016;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_ENDSESSION notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.235 Messages.WM_ENTERIDLE Constant

Represents the Windows message WM_ENTERIDLE.

Pascal

```
WM_ENTERIDLE = $0121;
```

C++

```
WM_ENTERIDLE = 0x0121;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_ENTERIDLE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.236 Messages.WM_ENTERMENULOOP Constant

Represents the Windows message WM_ENTERMENULOOP.

Pascal

```
WM_ENTERMENULOOP = $0211;
```

C++

```
WM_ENTERMENULOOP = 0x0211;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_ENTERMENULOOP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.237 Messages.WM_ENTERSIZEMOVE Constant

Represents the Windows message WM_ENTERSIZEMOVE.

Pascal

```
WM_ENTERSIZEMOVE = $0231;
```

C++

```
WM_ENTERSIZEMOVE = 0x0231;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_ENTERSIZEMOVE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.238 Messages.WM_ERASEBKGND Constant

Represents the Windows message WM_ERASEBKGND.

Pascal

```
WM_ERASEBKGND = $0014;
```

C++

```
WM_ERASEBKGND = 0x0014;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_ERASEBKGND notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.239 Messages.WM_EXITMENULOOP Constant

Represents the Windows message WM_EXITMENULOOP.

Pascal

```
WM_EXITMENULOOP = $0212;
```

C++

```
WM_EXITMENULOOP = 0x0212;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_EXITMENULOOP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.240 Messages.WM_EXITSIZEMOVE Constant

Represents the Windows message WM_EXITSIZEMOVE.

Pascal

```
WM_EXITSIZEMOVE = $0232;
```

C++

```
WM_EXITSIZEMOVE = 0x0232;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_EXITSIZEMOVE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.241 Messages.WM_FONTCHANGE Constant

Represents the Windows message WM_FONTCHANGE.

Pascal

```
WM_FONTCHANGE = $001D;
```

C++

```
WM_FONTCHANGE = 0x001D;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_FONTCHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.242 Messages.WM_GETDLGCODE Constant

Represents the Windows message WM_GETDLGCODE.

Pascal

```
WM_GETDLGCODE = $0087;
```

C++

```
WM_GETDLGCODE = 0x0087;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_GETDLGCODE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.243 Messages.WM_GETFONT Constant

Represents the Windows message WM_GETFONT.

Pascal

```
WM_GETFONT = $0031;
```

C++

```
WM_GETFONT = 0x0031;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_GETFONT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.244 Messages.WM_GETHOTKEY Constant

Represents the Windows message WM_GETHOTKEY.

Pascal

```
WM_GETHOTKEY = $0033;
```

C++

```
WM_GETHOTKEY = 0x0033;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_GETHOTKEY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.245 Messages.WM_GETICON Constant

Represents the Windows message WM_GETICON.

Pascal

```
WM_GETICON = $007F;
```

C++

```
WM_GETICON = 0x007F;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_GETICON notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.246 Messages.WM_GETMINMAXINFO Constant

Represents the Windows message WM_GETMINMAXINFO.

Pascal

```
WM_GETMINMAXINFO = $0024;
```

C++

```
WM_GETMINMAXINFO = 0x0024;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_GETMINMAXINFO

notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.247 Messages.WM_GETOBJECT Constant

Represents the Windows message WM_GETOBJECT.

Pascal

```
WM_GETOBJECT = $003D;
```

C++

```
WM_GETOBJECT = 0x003D;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_GETOBJECT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.248 Messages.WM_GETTEXT Constant

Represents the Windows message WM_GETTEXT.

Pascal

```
WM_GETTEXT = $000D;
```

C++

```
WM_GETTEXT = 0x000D;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_GETTEXT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.249 Messages.WM_GETTEXTLENGTH Constant

Represents the Windows message WM_GETTEXTLENGTH.

Pascal

```
WM_GETTEXTLENGTH = $000E;
```

C++

```
WM_GETTEXTLENGTH = 0x000E;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_GETTEXTLENGTH notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.250 Messages.WM_HANDHELDFIRST Constant

Represents the Windows message WM_HANDHELDFIRST.

Pascal

```
WM_HANDHELDFIRST = 856;
```

C++

```
WM_HANDHELDFIRST = 856;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.251 Messages.WM_HANDHELDLAST Constant

Represents the Windows message WM_HANDHELDLAST.

Pascal

```
WM_HANDHELDLAST = 863;
```

C++

```
WM_HANDHELDLAST = 863;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.252 Messages.WM_HELP Constant

Represents the Windows message WM_HELP.

Pascal

```
WM_HELP = $0053;
```

C++

```
WM_HELP = 0x0053;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_HELP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.253 Messages.WM_HOTKEY Constant

Represents the Windows message WM_HOTKEY.

Pascal

```
WM_HOTKEY = $0312;
```

C++

```
WM_HOTKEY = 0x0312;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_HOTKEY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.254 Messages.WM_HSCROLL Constant

Represents the Windows message WM_HSCROLL.

Pascal

```
WM_HSCROLL = $0114;
```

C++

```
WM_HSCROLL = 0x0114;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_HSCROLL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.255 Messages.WM_HSCROLLCLIPBOARD Constant

Represents the Windows message WM_HSCROLLCLIPBOARD.

Pascal

```
WM_HSCROLLCLIPBOARD = $030E;
```

C++

```
WM_HSCROLLCLIPBOARD = 0x030E;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_HSCROLLCLIPBOARD notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.256 Messages.WM_ICONERASEBKGND Constant

Represents the Windows message WM_ICONERASEBKGND.

Pascal

```
WM_ICONERASEBKGND = $0027;
```

C++

```
WM_ICONERASEBKGND = 0x0027;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_ICONERASEBKGND notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.257 Messages.WM_IME_CHAR Constant

Represents the Windows message WM_IME_CHAR.

Pascal

```
WM_IME_CHAR = $0286;
```

C++

```
WM_IME_CHAR = 0x0286;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_CHAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.258 Messages.WM_IME_COMPOSITION Constant

Represents the Windows message WM_IME_COMPOSITION.

Pascal

```
WM_IME_COMPOSITION = $010F;
```

C++

```
WM_IME_COMPOSITION = 0x010F;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_COMPOSITION notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.259 Messages.WM_IME_COMPOSITIONFULL Constant

Represents the Windows message WM_IME_COMPOSITIONFULL.

Pascal

```
WM_IME_COMPOSITIONFULL = $0284;
```

C++

```
WM_IME_COMPOSITIONFULL = 0x0284;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_COMPOSITIONFULL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.260 Messages.WM_IME_CONTROL Constant

Represents the Windows message WM_IME_CONTROL.

Pascal

```
WM_IME_CONTROL = $0283;
```

C++

```
WM_IME_CONTROL = 0x0283;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_CONTROL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.261 Messages.WM_IME_ENDCOMPOSITION Constant

Represents the Windows message WM_IME_ENDCOMPOSITION.

Pascal

```
WM_IME_ENDCOMPOSITION = $010E;
```

C++

```
WM_IME_ENDCOMPOSITION = 0x010E;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_ENDCOMPOSITION notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.262 Messages.WM_IME_KEYDOWN Constant

Represents the Windows message WM_IME_KEYDOWN.

Pascal

```
WM_IME_KEYDOWN = $0290;
```

C++

```
WM_IME_KEYDOWN = 0x0290;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_KEYDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.263 Messages.WM_IME_KEYLAST Constant

Represents the Windows message WM_IME_KEYLAST.

Pascal

```
WM_IME_KEYLAST = $010F;
```

C++

```
WM_IME_KEYLAST = 0x010F;
```

File

Messages

Description

This constant is used by Windows messaging. WM_IME_KEYLAST is obsolete for Win32-Based Applications.

See Also

SendMessage (MSDN)

Handling messages

35.4.264 Messages.WM_IME_KEYUP Constant

Represents the Windows message WM_IME_KEYUP.

Pascal

```
WM_IME_KEYUP = $0291;
```

C++

```
WM_IME_KEYUP = 0x0291;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_KEYUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.265 Messages.WM_IME_NOTIFY Constant

Represents the Windows message WM_IME_NOTIFY.

Pascal

```
WM_IME_NOTIFY = $0282;
```

C++

```
WM_IME_NOTIFY = 0x0282;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_NOTIFY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.266 Messages.WM_IME_REQUEST Constant

Represents the Windows message WM_IME_REQUEST.

Pascal

```
WM_IME_REQUEST = $0288;
```

C++

```
WM_IME_REQUEST = 0x0288;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_REQUEST notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.267 Messages.WM_IME_SELECT Constant

Represents the Windows message WM_IME_SELECT.

Pascal

```
WM_IME_SELECT = $0285;
```

C++

```
WM_IME_SELECT = 0x0285;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_SELECT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.268 Messages.WM_IME_SETCONTEXT Constant

Represents the Windows message WM_IME_SETCONTEXT.

Pascal

```
WM_IME_SETCONTEXT = $0281;
```

C++

```
WM_IME_SETCONTEXT = 0x0281;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_SETCONTEXT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.269 Messages.WM_IME_STARTCOMPOSITION Constant

Represents the Windows message WM_IME_STARTCOMPOSITION.

Pascal

```
WM_IME_STARTCOMPOSITION = $010D;
```

C++

```
WM_IME_STARTCOMPOSITION = 0x010D;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_IME_STARTCOMPOSITION notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.270 Messages.WM_INITDIALOG Constant

Represents the Windows message WM_INITDIALOG.

Pascal

```
WM_INITDIALOG = $0110;
```

C++

```
WM_INITDIALOG = 0x0110;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_INITDIALOG notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.271 Messages.WM_INITMENU Constant

Represents the Windows message WM_INITMENU.

Pascal

```
WM_INITMENU = $0116;
```

C++

```
WM_INITMENU = 0x0116;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_INITMENU notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.272 Messages.WM_INITMENUPOPUP Constant

Represents the Windows message WM_INITMENUPOPUP.

Pascal

```
WM_INITMENUPOPUP = $0117;
```

C++

```
WM_INITMENUPOPUP = 0x0117;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_INITMENUPOPUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.273 Messages.WM_INPUT Constant

Represents the Windows message WM_INPUT.

Pascal

```
WM_INPUT = $00FF;
```

C++

```
WM_INPUT = 0x00FF;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_INPUT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.274 Messages.WM_INPUTLANGCHANGE Constant

Represents the Windows message WM_INPUTLANGCHANGE.

Pascal

```
WM_INPUTLANGCHANGE = $0051;
```

C++

```
WM_INPUTLANGCHANGE = 0x0051;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_INPUTLANGCHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.275 Messages.WM_INPUTLANGCHANGEREQUEST Constant

Represents the Windows message WM_INPUTLANGCHANGEREQUEST.

Pascal

```
WM_INPUTLANGCHANGEREQUEST = $0050;
```

C++

```
WM_INPUTLANGCHANGEREQUEST = 0x0050;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_INPUTLANGCHANGEREQUEST notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.276 Messages.WM_KEYDOWN Constant

Represents the Windows message WM_KEYDOWN.

Pascal

```
WM_KEYDOWN = $0100;
```

C++

```
WM_KEYDOWN = 0x0100;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_KEYDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.277 Messages.WM_KEYFIRST Constant

Represents the Windows message WM_KEYFIRST.

Pascal

```
WM_KEYFIRST = $0100;
```

C++

```
WM_KEYFIRST = 0x0100;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.278 Messages.WM_KEYLAST Constant

Represents the Windows message WM_KEYLAST.

Pascal

```
WM_KEYLAST = $0109;
```

C++

```
WM_KEYLAST = 0x0109;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.279 Messages.WM_KEYUP Constant

Represents the Windows message WM_KEYUP.

Pascal

```
WM_KEYUP = $0101;
```

C++

```
WM_KEYUP = 0x0101;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_KEYUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.280 Messages.WM_KILLFOCUS Constant

Represents the Windows message WM_KILLFOCUS.

Pascal

```
WM_KILLFOCUS = $0008;
```

C++

```
WM_KILLFOCUS = 0x0008;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_KILLFOCUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.281 Messages.WM_LBUTTONDBLCLK Constant

Represents the Windows message WM_LBUTTONDBLCLK.

Pascal

```
WM_LBUTTONDBLCLK = $0203;
```

C++

```
WM_LBUTTONDBLCLK = 0x0203;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_LBUTTONDBLCLK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.282 Messages.WM_LBUTTONDOWN Constant

Represents the Windows message WM_LBUTTONDOWN.

Pascal

```
WM_LBUTTONDOWN = $0201;
```

C++

```
WM_LBUTTONDOWN = 0x0201;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_LBUTTONDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.283 Messages.WM_LBUTTONUP Constant

Represents the Windows message WM_LBUTTONUP.

Pascal

```
WM_LBUTTONUP = $0202;
```

C++

```
WM_LBUTTONUP = 0x0202;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_LBUTTONUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.284 Messages.WM_MBUTTONDBLCLK Constant

Represents the Windows message WM_MBUTTONDBLCLK.

Pascal

```
WM_MBUTTONDBLCLK = $0209;
```

C++

```
WM_MBUTTONDBLCLK = 0x0209;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MBUTTONDBLCLK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.285 Messages.WM_MBUTTONDOWN Constant

Represents the Windows message WM_MBUTTONDOWN.

Pascal

```
WM_MBUTTONDOWN = $0207;
```

C++

```
WM_MBUTTONDOWN = 0x0207;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MBUTTONDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.286 Messages.WM_MBUTTONUP Constant

Represents the Windows message WM_MBUTTONUP.

Pascal

```
WM_MBUTTONUP = $0208;
```

C++

```
WM_MBUTTONUP = 0x0208;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MBUTTONUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.287 Messages.WM_MDIACTIVATE Constant

Represents the Windows message WM_MDIACTIVATE.

Pascal

```
WM_MDIACTIVATE = $0222;
```

C++

```
WM_MDIACTIVATE = 0x0222;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDIACTIVATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.288 Messages.WM_MDICASCADE Constant

Represents the Windows message WM_MDICASCADE.

Pascal

```
WM_MDICASCADE = $0227;
```

C++

```
WM_MDICASCADE = 0x0227;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDICASCADE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.289 Messages.WM_MDICREATE Constant

Represents the Windows message WM_MDICREATE.

Pascal

```
WM_MDICREATE = $0220;
```

C++

```
WM_MDICREATE = 0x0220;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDICREATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.290 Messages.WM_MDIDESTROY Constant

Represents the Windows message WM_MDIDESTROY.

Pascal

```
WM_MDIDESTROY = $0221;
```

C++

```
WM_MDIDESTROY = 0x0221;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDIDESTROY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.291 Messages.WM_MDIGETACTIVE Constant

Represents the Windows message WM_MDIGETACTIVE.

Pascal

```
WM_MDIGETACTIVE = $0229;
```

C++

```
WM_MDIGETACTIVE = 0x0229;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDIGETACTIVE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.292 Messages.WM_MDIICONARRANGE Constant

Represents the Windows message WM_MDIICONARRANGE.

Pascal

```
WM_MDIICONARRANGE = $0228;
```

C++

```
WM_MDIICONARRANGE = 0x0228;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDIICONARRANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.293 Messages.WM_MDIMAXIMIZE Constant

Represents the Windows message WM_MDIMAXIMIZE.

Pascal

```
WM_MDIMAXIMIZE = $0225;
```

C++

```
WM_MDIMAXIMIZE = 0x0225;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDIMAXIMIZE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.294 Messages.WM_MDINEXT Constant

Represents the Windows message WM_MDINEXT.

Pascal

```
WM_MDINEXT = $0224;
```

C++

```
WM_MDINEXT = 0x0224;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDINEXT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.295 Messages.WM_MDIREFRESHMENU Constant

Represents the Windows message WM_MDIREFRESHMENU.

Pascal

```
WM_MDIREFRESHMENU = $0234;
```

C++

```
WM_MDIREFRESHMENU = 0x0234;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDIREFRESHMENU notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.296 Messages.WM_MDIRESTORE Constant

Represents the Windows message WM_MDIRESTORE.

Pascal

```
WM_MDIRESTORE = $0223;
```

C++

```
WM_MDIRESTORE = 0x0223;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDIRESTORE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.297 Messages.WM_MDISETMENU Constant

Represents the Windows message WM_MDISETMENU.

Pascal

```
WM_MDISETMENU = $0230;
```

C++

```
WM_MDISETMENU = 0x0230;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDISETMENU notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.298 Messages.WM_MDITILE Constant

Represents the Windows message WM_MDITILE.

Pascal

```
WM_MDITILE = $0226;
```

C++

```
WM_MDITILE = 0x0226;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MDITILE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.299 Messages.WM_MEASUREITEM Constant

Represents the Windows message WM_MEASUREITEM.

Pascal

```
WM_MEASUREITEM = $002C;
```

C++

```
WM_MEASUREITEM = 0x002C;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MEASUREITEM notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.300 Messages.WM_MENUCHAR Constant

Represents the Windows message WM_MENUCHAR.

Pascal

```
WM_MENUCHAR = $0120;
```

C++

```
WM_MENUCHAR = 0x0120;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MENUCHAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.301 Messages.WM_MENUCOMMAND Constant

Represents the Windows message WM_MENUCOMMAND.

Pascal

```
WM_MENUCOMMAND = $0126;
```

C++

```
WM_MENUCOMMAND = 0x0126;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MENUCOMMAND notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.302 Messages.WM_MENUDRAG Constant

Represents the Windows message WM_MENUDRAG.

Pascal

```
WM_MENUDRAG = $0123;
```

C++

```
WM_MENUDRAG = 0x0123;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MENUDRAG notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.303 Messages.WM_MENUGETOBJECT Constant

Represents the Windows message WM_MENUGETOBJECT.

Pascal

```
WM_MENUGETOBJECT = $0124;
```

C++

```
WM_MENUGETOBJECT = 0x0124;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MENUGETOBJECT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.304 Messages.WM_MENURBUTTONUP Constant

Represents the Windows message WM_MENURBUTTONUP.

Pascal

```
WM_MENURBUTTONUP = $0122;
```

C++

```
WM_MENURBUTTONUP = 0x0122;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MENURBUTTONUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.305 Messages.WM_MENUSELECT Constant

Represents the Windows message WM_MENUSELECT.

Pascal

```
WM_MENUSELECT = $011F;
```

C++

```
WM_MENUSELECT = 0x011F;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MENUSELECT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.306 Messages.WM_MOUSEACTIVATE Constant

Represents the Windows message WM_MOUSEACTIVATE.

Pascal

```
WM_MOUSEACTIVATE = $0021;
```

C++

```
WM_MOUSEACTIVATE = 0x0021;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MOUSEACTIVATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.307 Messages.WM_MOUSEFIRST Constant

Represents the Windows message WM_MOUSEFIRST.

Pascal

```
WM_MOUSEFIRST = $0200;
```

C++

```
WM_MOUSEFIRST = 0x0200;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.308 Messages.WM_MOUSEHOVER Constant

Represents the Windows message WM_MOUSEHOVER.

Pascal

```
WM_MOUSEHOVER = $02A1;
```

C++

```
WM_MOUSEHOVER = 0x02A1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MOUSEHOVER notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.309 Messages.WM_MOUSELAST Constant

Represents the Windows message WM_MOUSELAST.

Pascal

```
WM_MOUSELAST = $020A;
```

C++

```
WM_MOUSELAST = 0x020A;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.310 Messages.WM_MOUSELEAVE Constant

Represents the Windows message WM_MOUSELEAVE.

Pascal

```
WM_MOUSELEAVE = $02A3;
```

C++

```
WM_MOUSELEAVE = 0x02A3;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MOUSELEAVE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.311 Messages.WM_MOUSEMOVE Constant

Represents the Windows message WM_MOUSEMOVE.

Pascal

```
WM_MOUSEMOVE = $0200;
```

C++

```
WM_MOUSEMOVE = 0x0200;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MOUSEMOVE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.312 Messages.WM_MOUSEWHEEL Constant

Represents the Windows message WM_MOUSEWHEEL.

Pascal

```
WM_MOUSEWHEEL = $020A;
```

C++

```
WM_MOUSEWHEEL = 0x020A;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MOUSEWHEEL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.313 Messages.WM_MOVE Constant

Represents the Windows message WM_MOVE.

Pascal

```
WM_MOVE = $0003;
```

C++

```
WM_MOVE = 0x0003;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MOVE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.314 Messages.WM_MOVING Constant

Represents the Windows message WM_MOVING.

Pascal

```
WM_MOVING = 534;
```

C++

```
WM_MOVING = 534;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_MOVING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.315 Messages.WM_NCACTIVATE Constant

Represents the Windows message WM_NCACTIVATE.

Pascal

```
WM_NCACTIVATE = $0086;
```

C++

```
WM_NCACTIVATE = 0x0086;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCACTIVATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.316 Messages.WM_NCCALCSIZE Constant

Represents the Windows message WM_NCCALCSIZE.

Pascal

```
WM_NCCALCSIZE = $0083;
```

C++

```
WM_NCCALCSIZE = 0x0083;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCCALCSIZE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.317 Messages.WM_NCCREATE Constant

Represents the Windows message WM_NCCREATE.

Pascal

```
WM_NCCREATE = $0081;
```

C++

```
WM_NCCREATE = 0x0081;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCCREATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.318 Messages.WM_NCDESTROY Constant

Represents the Windows message WM_NCDESTROY.

Pascal

```
WM_NCDESTROY = $0082;
```

C++

```
WM_NCDESTROY = 0x0082;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCDESTROY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.319 Messages.WM_NCHITTEST Constant

Represents the Windows message WM_NCHITTEST.

Pascal

```
WM_NCHITTEST = $0084;
```

C++

```
WM_NCHITTEST = 0x0084;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCHITTEST notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.320 Messages.WM_NCLBUTTONDBLCLK Constant

Represents the Windows message WM_NCLBUTTONDBLCLK.

Pascal

```
WM_NCLBUTTONDBLCLK = $00A3;
```

C++

```
WM_NCLBUTTONDBLCLK = 0x00A3;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCLBUTTONDBLCLK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.321 Messages.WM_NCLBUTTONDOWN Constant

Represents the Windows message WM_NCLBUTTONDOWN.

Pascal

```
WM_NCLBUTTONDOWN = $00A1;
```

C++

```
WM_NCLBUTTONDOWN = 0x00A1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCLBUTTONDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.322 Messages.WM_NCLBUTTONUP Constant

Represents the Windows message WM_NCLBUTTONUP.

Pascal

```
WM_NCLBUTTONUP = $00A2;
```

C++

```
WM_NCLBUTTONUP = 0x00A2;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCLBUTTONUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.323 Messages.WM_NCMBUTTONDBLCLK Constant

Represents the Windows message WM_NCMBUTTONDBLCLK.

Pascal

```
WM_NCMBUTTONDBLCLK = $00A9;
```

C++

```
WM_NCMBUTTONDBLCLK = 0x00A9;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCMBUTTONDBLCLK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.324 Messages.WM_NCMBUTTONDOWN Constant

Represents the Windows message WM_NCMBUTTONDOWN.

Pascal

```
WM_NCMBUTTONDOWN = $00A7;
```

C++

```
WM_NCMBUTTONDOWN = 0x00A7;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCMBUTTONDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.325 Messages.WM_NCMBUTTONUP Constant

Represents the Windows message WM_NCMBUTTONUP.

Pascal

```
WM_NCMBUTTONUP = $00A8;
```

C++

```
WM_NCMBUTTONUP = 0x00A8;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCMBUTTONUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.326 Messages.WM_NCMOUSEHOVER Constant

Represents the Windows message WM_NCMOUSEHOVER.

Pascal

```
WM_NCMOUSEHOVER = $02A0;
```

C++

```
WM_NCMOUSEHOVER = 0x02A0;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCMOUSEHOVER notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.327 Messages.WM_NCMOUSELEAVE Constant

Represents the Windows message WM_NCMOUSELEAVE.

Pascal

```
WM_NCMOUSELEAVE = $02A2;
```

C++

```
WM_NCMOUSELEAVE = 0x02A2;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCMOUSELEAVE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.328 Messages.WM_NCMOUSEMOVE Constant

Represents the Windows message WM_NCMOUSEMOVE.

Pascal

```
WM_NCMOUSEMOVE = $00A0;
```

C++

```
WM_NCMOUSEMOVE = 0x00A0;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCMOUSEMOVE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.329 Messages.WM_NCPAINT Constant

Represents the Windows message WM_NCPAINT.

Pascal

```
WM_NCPAINT = $0085;
```

C++

```
WM_NCPAINT = 0x0085;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCPAINT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.330 Messages.WM_NCRBUTTONDBLCLK Constant

Represents the Windows message WM_NCRBUTTONDBLCLK.

Pascal

```
WM_NCRBUTTONDBLCLK = $00A6;
```

C++

```
WM_NCRBUTTONDBLCLK = 0x00A6;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCRBUTTONDBLCLK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.331 Messages.WM_NCRBUTTONDOWN Constant

Represents the Windows message WM_NCRBUTTONDOWN.

Pascal

```
WM_NCRBUTTONDOWN = $00A4;
```

C++

```
WM_NCRBUTTONDOWN = 0x00A4;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCRBUTTONDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.332 Messages.WM_NCRBUTTONUP Constant

Represents the Windows message WM_NCRBUTTONUP.

Pascal

```
WM_NCRBUTTONUP = $00A5;
```

C++

```
WM_NCRBUTTONUP = 0x00A5;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCRBUTTONUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.333 Messages.WM_NCXBUTTONDBLCLK Constant

Represents the Windows message WM_NCXBUTTONDBLCLK.

Pascal

```
WM_NCXBUTTONDBLCLK = $00AD;
```

C++

```
WM_NCXBUTTONDBLCLK = 0x00AD;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCXBUTTONDBLCLK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.334 Messages.WM_NCXBUTTONDOWN Constant

Represents the Windows message WM_NCXBUTTONDOWN.

Pascal

```
WM_NCXBUTTONDOWN = $00AB;
```

C++

```
WM_NCXBUTTONDOWN = 0x00AB;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCXBUTTONDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.335 Messages.WM_NCXBUTTONUP Constant

Represents the Windows message WM_NCXBUTTONUP.

Pascal

```
WM_NCXBUTTONUP = $00AC;
```

C++

```
WM_NCXBUTTONUP = 0x00AC;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NCXBUTTONUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.336 Messages.WM_NEXTDLGCTL Constant

Represents the Windows message WM_NEXTDLGCTL.

Pascal

```
WM_NEXTDLGCTL = $0028;
```

C++

```
WM_NEXTDLGCTL = 0x0028;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NEXTDLGCTL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.337 Messages.WM_NEXTMENU Constant

Represents the Windows message WM_NEXTMENU.

Pascal

```
WM_NEXTMENU = $0213;
```

C++

```
WM_NEXTMENU = 0x0213;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NEXTMENU notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.338 Messages.WM_NOTIFY Constant

Represents the Windows message WM_NOTIFY.

Pascal

```
WM_NOTIFY = $004E;
```

C++

```
WM_NOTIFY = 0x004E;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NOTIFY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.339 Messages.WM_NOTIFYFORMAT Constant

Represents the Windows message WM_NOTIFYFORMAT.

Pascal

```
WM_NOTIFYFORMAT = $0055;
```

C++

```
WM_NOTIFYFORMAT = 0x0055;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NOTIFYFORMAT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.340 Messages.WM_NULL Constant

Represents the Windows message WM_NULL.

Pascal

```
WM_NULL = $0000;
```

C++

```
WM_NULL = 0x0000;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_NULL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.341 Messages.WM_PAINT Constant

Represents the Windows message WM_PAINT.

Pascal

```
WM_PAINT = $000F;
```

C++

```
WM_PAINT = 0x000F;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_PAINT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.342 Messages.WM_PAINTCLIPBOARD Constant

Represents the Windows message WM_PAINTCLIPBOARD.

Pascal

```
WM_PAINTCLIPBOARD = $0309;
```

C++

```
WM_PAINTCLIPBOARD = 0x0309;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_PAINTCLIPBOARD notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.343 Messages.WM_PAINTICON Constant

Represents the Windows message WM_PAINTICON.

Pascal

```
WM_PAINTICON = $0026;
```

C++

```
WM_PAINTICON = 0x0026;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_PAINTICON notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.344 Messages.WM_PALETTECHANGED Constant

Represents the Windows message WM_PALETTECHANGED.

Pascal

```
WM_PALETTECHANGED = $0311;
```

C++

```
WM_PALETTECHANGED = 0x0311;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_PALETTECHANGED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.345 Messages.WM_PALETTEISCHANGING Constant

Represents the Windows message WM_PALETTEISCHANGING.

Pascal

```
WM_PALETTEISCHANGING = $0310;
```

C++

```
WM_PALETTEISCHANGING = 0x0310;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_PALETTEISCHANGING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.346 Messages.WM_PARENTNOTIFY Constant

Represents the Windows message WM_PARENTNOTIFY.

Pascal

```
WM_PARENTNOTIFY = $0210;
```

C++

```
WM_PARENTNOTIFY = 0x0210;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_PARENTNOTIFY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.347 Messages.WM_PASTE Constant

Represents the Windows message WM_PASTE.

Pascal

```
WM_PASTE = $0302;
```

C++

```
WM_PASTE = 0x0302;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_PASTE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.348 Messages.WM_PENWINFIRST Constant

Represents the Windows message WM_PENWINFIRST.

Pascal

```
WM_PENWINFIRST = $0380;
```

C++

```
WM_PENWINFIRST = 0x0380;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.349 Messages.WM_PENWINLAST Constant

Represents the Windows message WM_PENWINLAST.

Pascal

```
WM_PENWINLAST = $038F;
```

C++

```
WM_PENWINLAST = 0x038F;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.350 Messages.WM_POWER Constant

Represents the Windows message WM_POWER.

Pascal

```
WM_POWER = $0048;
```

C++

```
WM_POWER = 0x0048;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_POWER notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.351 Messages.WM_POWERBROADCAST Constant

Represents the Windows message WM_POWERBROADCAST.

Pascal

```
WM_POWERBROADCAST = 536;
```

C++

```
WM_POWERBROADCAST = 536;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_POWERBROADCAST

notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.352 Messages.WM_PRINT Constant

Represents the Windows message WM_PRINT.

Pascal

```
WM_PRINT = 791;
```

C++

```
WM_PRINT = 791;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_PRINT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.353 Messages.WM_PRINTCLIENT Constant

Represents the Windows message WM_PRINTCLIENT.

Pascal

```
WM_PRINTCLIENT = 792;
```

C++

```
WM_PRINTCLIENT = 792;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_PRINTCLIENT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.354 Messages.WM_QUERYDRAGICON Constant

Represents the Windows message WM_QUERYDRAGICON.

Pascal

```
WM_QUERYDRAGICON = $0037;
```

C++

```
WM_QUERYDRAGICON = 0x0037;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_QUERYDRAGICON notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.355 Messages.WM_QUERYENDSESSION Constant

Represents the Windows message WM_QUERYENDSESSION.

Pascal

```
WM_QUERYENDSESSION = $0011;
```

C++

```
WM_QUERYENDSESSION = 0x0011;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_QUERYENDSESSION notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.356 Messages.WM_QUERYNEWPALETTE Constant

Represents the Windows message WM_QUERYNEWPALETTE.

Pascal

```
WM_QUERYNEWPALETTE = $030F;
```

C++

```
WM_QUERYNEWPALETTE = 0x030F;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_QUERYNEWPALETTE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.357 Messages.WM_QUERYOPEN Constant

Represents the Windows message WM_QUERYOPEN.

Pascal

```
WM_QUERYOPEN = $0013;
```

C++

```
WM_QUERYOPEN = 0x0013;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_QUERYOPEN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.358 Messages.WM_QUERYUISTATE Constant

Represents the Windows message WM_QUERYUISTATE.

Pascal

```
WM_QUERYUISTATE = $0129;
```

C++

```
WM_QUERYUISTATE = 0x0129;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_QUERYUISTATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.359 Messages.WM_QUEUESYNC Constant

Represents the Windows message WM_QUEUESYNC.

Pascal

```
WM_QUEUESYNC = $0023;
```

C++

```
WM_QUEUESYNC = 0x0023;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_QUEUESYNC notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.360 Messages.WM_QUIT Constant

Represents the Windows message WM_QUIT.

Pascal

```
WM_QUIT = $0012;
```

C++

```
WM_QUIT = 0x0012;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_QUIT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.361 Messages.WM_RBUTTONDBLCLK Constant

Represents the Windows message WM_RBUTTONDBLCLK.

Pascal

```
WM_RBUTTONDBLCLK = $0206;
```

C++

```
WM_RBUTTONDBLCLK = 0x0206;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_RBUTTONDBLCLK notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.362 Messages.WM_RBUTTONDOWN Constant

Represents the Windows message WM_RBUTTONDOWN.

Pascal

```
WM_RBUTTONDOWN = $0204;
```

C++

```
WM_RBUTTONDOWN = 0x0204;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_RBUTTONDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.363 Messages.WM_RBUTTONUP Constant

Represents the Windows message WM_RBUTTONUP.

Pascal

```
WM_RBUTTONUP = $0205;
```

C++

```
WM_RBUTTONUP = 0x0205;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_RBUTTONUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.364 Messages.WM_RENDERALLFORMATS Constant

Represents the Windows message WM_RENDERALLFORMATS.

Pascal

```
WM_RENDERALLFORMATS = $0306;
```

C++

```
WM_RENDERALLFORMATS = 0x0306;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_RENDERALLFORMATS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.365 Messages.WM_RENDERFORMAT Constant

Represents the Windows message WM_RENDERFORMAT.

Pascal

```
WM_RENDERFORMAT = $0305;
```

C++

```
WM_RENDERFORMAT = 0x0305;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_RENDERFORMAT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.366 Messages.WM_SETCURSOR Constant

Represents the Windows message WM_SETCURSOR.

Pascal

```
WM_SETCURSOR = $0020;
```

C++

```
WM_SETCURSOR = 0x0020;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SETCURSOR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.367 Messages.WM_SETFOCUS Constant

Represents the Windows message WM_SETFOCUS.

Pascal

```
WM_SETFOCUS = $0007;
```

C++

```
WM_SETFOCUS = 0x0007;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SETFOCUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.368 Messages.WM_SETFONT Constant

Represents the Windows message WM_SETFONT.

Pascal

```
WM_SETFONT = $0030;
```

C++

```
WM_SETFONT = 0x0030;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SETFONT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.369 Messages.WM_SETHOTKEY Constant

Represents the Windows message WM_SETHOTKEY.

Pascal

```
WM_SETHOTKEY = $0032;
```

C++

```
WM_SETHOTKEY = 0x0032;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SETHOTKEY notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.370 Messages.WM_SETICON Constant

Represents the Windows message WM_SETICON.

Pascal

```
WM_SETICON = $0080;
```

C++

```
WM_SETICON = 0x0080;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SETICON notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.371 Messages.WM_SETREDRAW Constant

Represents the Windows message WM_SETREDRAW.

Pascal

```
WM_SETREDRAW = $000B;
```

C++

```
WM_SETREDRAW = 0x000B;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SETREDRAW notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.372 Messages.WM_SETTEXT Constant

Represents the Windows message WM_SETTEXT.

Pascal

```
WM_SETTEXT = $000C;
```

C++

```
WM_SETTEXT = 0x000C;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SETTEXT notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.373 Messages.WM_SETTINGCHANGE Constant

Represents the Windows message WM_SETTINGCHANGE.

Pascal

```
WM_SETTINGCHANGE = WM_WININICHANGE;
```

C++

```
WM_SETTINGCHANGE = WM_WININICHANGE;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SETTINGCHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.374 Messages.WM_SHOWWINDOW Constant

Represents the Windows message WM_SHOWWINDOW.

Pascal

```
WM_SHOWWINDOW = $0018;
```

C++

```
WM_SHOWWINDOW = 0x0018;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SHOWWINDOW notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.375 Messages.WM_SIZE Constant

Represents the Windows message WM_SIZE.

Pascal

```
WM_SIZE = $0005;
```

C++

```
WM_SIZE = 0x0005;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SIZE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.376 Messages.WM_SIZECLIPBOARD Constant

Represents the Windows message WM_SIZECLIPBOARD.

Pascal

```
WM_SIZECLIPBOARD = $030B;
```

C++

```
WM_SIZECLIPBOARD = 0x030B;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SIZECLIPBOARD notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.377 Messages.WM_SIZING Constant

Represents the Windows message WM_SIZING.

Pascal

```
WM_SIZING = 532;
```

C++

```
WM_SIZING = 532;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SIZING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.378 Messages.WM_SPOOLERSTATUS Constant

Represents the Windows message WM_SPOOLERSTATUS.

Pascal

```
WM_SPOOLERSTATUS = $002A;
```

C++

```
WM_SPOOLERSTATUS = 0x002A;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SPOOLERSTATUS notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.379 Messages.WM_STYLECHANGED Constant

Represents the Windows message WM_STYLECHANGED.

Pascal

```
WM_STYLECHANGED = $007D;
```

C++

```
WM_STYLECHANGED = 0x007D;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_STYLECHANGED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.380 Messages.WM_STYLECHANGING Constant

Represents the Windows message WM_STYLECHANGING.

Pascal

```
WM_STYLECHANGING = $007C;
```

C++

```
WM_STYLECHANGING = 0x007C;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_STYLECHANGING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.381 Messages.WM_SYSCHAR Constant

Represents the Windows message WM_SYSCHAR.

Pascal

```
WM_SYSCHAR = $0106;
```

C++

```
WM_SYSCHAR = 0x0106;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SYSCHAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.382 Messages.WM_SYSCOLORCHANGE Constant

Represents the Windows message WM_SYSCOLORCHANGE.

Pascal

```
WM_SYSCOLORCHANGE = $0015;
```

C++

```
WM_SYSCOLORCHANGE = 0x0015;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SYSCOLORCHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.383 Messages.WM_SYSCOMMAND Constant

Represents the Windows message WM_SYSCOMMAND.

Pascal

```
WM_SYSCOMMAND = $0112;
```

C++

```
WM_SYSCOMMAND = 0x0112;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SYSCOMMAND notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.384 Messages.WM_SYSDEADCHAR Constant

Represents the Windows message WM_SYSDEADCHAR.

Pascal

```
WM_SYSDEADCHAR = $0107;
```

C++

```
WM_SYSDEADCHAR = 0x0107;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SYSDEADCHAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.385 Messages.WM_SYSKEYDOWN Constant

Represents the Windows message WM_SYSKEYDOWN.

Pascal

```
WM_SYSKEYDOWN = $0104;
```

C++

```
WM_SYSKEYDOWN = 0x0104;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SYSKEYDOWN notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.386 Messages.WM_SYSKEYUP Constant

Represents the Windows message WM_SYSKEYUP.

Pascal

```
WM_SYSKEYUP = $0105;
```

C++

```
WM_SYSKEYUP = 0x0105;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_SYSKEYUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.387 Messages.WM_SYSTEMERROR Constant

Represents the Windows message WM_SYSTEMERROR.

Pascal

```
WM_SYSTEMERROR = $0017;
```

C++

```
WM_SYSTEMERROR = 0x0017;
```

File

Messages

Description

This constant is used by Windows messaging. WM_SYSTEMERROR is obsolete for Win32-Based Applications.

See Also

SendMessage (MSDN)

Handling messages

35.4.388 Messages.WM_TABLET_FIRST Constant

Represents the Windows message WM_TABLET_FIRST.

Pascal

```
WM_TABLET_FIRST = $02C0;
```

C++

```
WM_TABLET_FIRST = 0x02C0;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.389 Messages.WM_TABLET_LAST Constant

Represents the Windows message WM_TABLET_LAST.

Pascal

```
WM_TABLET_LAST = $02DF;
```

C++

```
WM_TABLET_LAST = 0x02DF;
```

File

Messages

Description

This constant is used by Windows messaging. Functions like GetMessage or PeekMessage use this constant for filtering purposes.

See Also

GetMessage (MSDN)

PeekMessage (MSDN)

Handling messages

35.4.390 Messages.WM_TCARD Constant

Represents the Windows message WM_TCARD.

Pascal

```
WM_TCARD = $0052;
```

C++

```
WM_TCARD = 0x0052;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_TCARD notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.391 Messages.WM_THEMECHANGED Constant

Represents the Windows message WM_THEMECHANGED.

Pascal

```
WM_THEMECHANGED = $031A;
```

C++

```
WM_THEMECHANGED = 0x031A;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_THEMECHANGED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.392 Messages.WM_TIMECHANGE Constant

Represents the Windows message WM_TIMECHANGE.

Pascal

```
WM_TIMECHANGE = $001E;
```

C++

```
WM_TIMECHANGE = 0x001E;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_TIMECHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.393 Messages.WM_TIMER Constant

Represents the Windows message WM_TIMER.

Pascal

```
WM_TIMER = $0113;
```

C++

```
WM_TIMER = 0x0113;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_TIMER notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.394 Messages.WM_UNDO Constant

Represents the Windows message WM_UNDO.

Pascal

```
WM_UNDO = $0304;
```

C++

```
WM_UNDO = 0x0304;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_UNDO notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.395 Messages.WM_UNICHAR Constant

Represents the Windows message WM_UNICHAR.

Pascal

```
WM_UNICHAR = $0109;
```

C++

```
WM_UNICHAR = 0x0109;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_UNICHAR notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.396 Messages.WM_UNINITMENUPOPUP Constant

Represents the Windows message WM_UNINITMENUPOPUP.

Pascal

```
WM_UNINITMENUPOPUP = $0125;
```

C++

```
WM_UNINITMENUPOPUP = 0x0125;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_UNINITMENUPOPUP notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.397 Messages.WM_UPDATEUISTATE Constant

Represents the Windows message WM_UPDATEUISTATE.

Pascal

```
WM_UPDATEUISTATE = $0128;
```

C++

```
WM_UPDATEUISTATE = 0x0128;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_UPDATEUISTATE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.398 Messages.WM_USER Constant

Represents the Windows message WM_USER.

Pascal

```
WM_USER = $0400;
```

C++

```
WM_USER = 0x0400;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_USER notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.399 Messages.WM_USERCHANGED Constant

Represents the Windows message WM_USERCHANGED.

Pascal

```
WM_USERCHANGED = $0054;
```

C++

```
WM_USERCHANGED = 0x0054;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_USERCHANGED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.400 Messages.WM_VKEYTOITEM Constant

Represents the Windows message WM_VKEYTOITEM.

Pascal

```
WM_VKEYTOITEM = $002E;
```

C++

```
WM_VKEYTOITEM = 0x002E;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_VKEYTOITEM notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.401 Messages.WM_VSCROLL Constant

Represents the Windows message WM_VSCROLL.

Pascal

```
WM_VSCROLL = $0115;
```

C++

```
WM_VSCROLL = 0x0115;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_VSCROLL notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.402 Messages.WM_VSCROLLCLIPBOARD Constant

Represents the Windows message WM_VSCROLLCLIPBOARD.

Pascal

```
WM_VSCROLLCLIPBOARD = $030A;
```

C++

```
WM_VSCROLLCLIPBOARD = 0x030A;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_VSCROLLCLIPBOARD notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.403 Messages.WM_WINDOWPOSCHANGED Constant

Represents the Windows message WM_WINDOWPOSCHANGED.

Pascal

```
WM_WINDOWPOSCHANGED = $0047;
```

C++

```
WM_WINDOWPOSCHANGED = 0x0047;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_WINDOWPOSCHANGED notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.404 Messages.WM_WINDOWPOSCHANGING Constant

Represents the Windows message WM_WINDOWPOSCHANGING.

Pascal

```
WM_WINDOWPOSCHANGING = $0046;
```

C++

```
WM_WINDOWPOSCHANGING = 0x0046;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_WINDOWPOSCHANGING notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.405 Messages.WM_WININICHANGE Constant

Represents the Windows message WM_WININICHANGE.

Pascal

```
WM_WININICHANGE = $001A;
```

C++

```
WM_WININICHANGE = 0x001A;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_WININICHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

35.4.406 Messages.WM_WTSSESSION_CHANGE Constant

Represents the Windows message WM_WTSSESSION_CHANGE.

Pascal

```
WM_WTSSESSION_CHANGE = $02B1;
```

C++

```
WM_WTSSESSION_CHANGE = 0x02B1;
```

File

Messages

Description

This constant is used by Windows messaging. For more information see the MSDN article for WM_WTSSESSION_CHANGE notification.

See Also

SendMessage (MSDN)

Handling messages

36 MidasCon Namespace

This is namespace MidasCon.

Topics

Name	Description
Classes (see page 1782)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 1789)	The following table lists structs, records, enums in this documentation.

Classes

	Name	Description
	TMidasConnection (see page 1782)	TMidasConnection handles a server connection for the client of a multi-tiered application.
	TRemoteServer (see page 1787)	TRemoteServer handles a DCOM connection to a remote server in a client application that is part of a multi-tiered application.

Structs, Records, Enums

	Name	Description
	TConnectType (see page 1789)	This is record MidasCon.TConnectType.

36.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TMidasConnection (see page 1782)	TMidasConnection handles a server connection for the client of a multi-tiered application.
	TRemoteServer (see page 1787)	TRemoteServer handles a DCOM connection to a remote server in a client application that is part of a multi-tiered application.

36.1.1 TMidasConnection Class

TMidasConnection handles a server connection for the client of a multi-tiered application.

Class Hierarchy

Pascal

```
TMidasConnection = class(TDCOMConnection);
```

C++

```
class TMidasConnection : public TDCOMConnection;
```

File

MidasCon

Description

TMidasConnection is provided for backward compatibility. When writing client applications that connect to application servers using DCOM, use TDCOMConnection (see page 1284) instead. When writing client applications that connect to application servers using sockets, use TSocketConnection instead. When writing client applications that connect to application servers using HTTP, use TWebConnection instead.

See Also

TClientDataSet

TDCOMConnection (see page 1284)

TSocketConnection

TWebConnection

36.1.1.1 TMidasConnection Members

The following tables list the members exposed by TMidasConnection.

Protected Methods

	Name	Description
	DoConnect (see page 1784)	Connects to an application server.
	DoDisconnect (see page 1785)	Drops a connection to an application server.
	GetConnected (see page 1293)	Returns the value of the Connected (see page 1296) property.
	GetServerCLSID (see page 1293)	Returns the GUID for the application server.
	Notification (see page 1294)	Responds when other objects are created or destroyed.
	SetConnected (see page 1280)	Sets the Connected (see page 1296) property.
	SetConnected (see page 1295)	Sets the Connected (see page 1296) property.
	SetConnectType (see page 1785)	This is SetConnectType, a member of class TMidasConnection.
	SetUseBroker (see page 1785)	This is SetUseBroker, a member of class TMidasConnection.

Protected Properties

	Name	Description
	ObjectBroker (see page 1300)	Specifies an Object Broker that allows the connection component to choose from several application servers.

Public Methods

	Name	Description
	Create (see page 1783)	Creates and initializes an instance of TMidasConnection (see page 1782).
	GetServer (see page 1293)	Returns the IAppServer interface of the application server.
	GetServerList (see page 1294)	Returns a list of remote data modules to which the component can connect.

Published Events

	Name	Description
	AfterConnect (see page 1302)	Occurs after a connection is established.
	AfterDisconnect (see page 1302)	Occurs after the connection closes.
	BeforeConnect (see page 1303)	Occurs immediately before establishing a connection.
	BeforeDisconnect (see page 1303)	Occurs immediately before the connection closes.
	OnGetUsername (see page 1304)	Occurs immediately before the remote login dialog.
	OnLogin (see page 1304)	Occurs when a communication channel opens to the server.

Published Properties

	Name	Description
	ComputerName (see page 1287)	Specifies the name of the machine on which the application server is located.
	Connected (see page 1296)	Indicates whether the TDispatchConnection (see page 1290) object is connected to an application server.
	ConnectType (see page 1786)	Specifies the protocol to use to establish the connection to the application server.
	LoginPrompt (see page 1296)	Specifies whether a login dialog appears immediately before opening a new connection.
	ObjectBroker (see page 1288)	Specifies an Object Broker that allows the connection component to choose from several application servers.
	ServerGUID (see page 1301)	Specifies the GUID of the application server with which to connect.
	ServerName (see page 1301)	Specifies the name of the application server with which to connect.
	ServerPort (see page 1786)	Specifies the port used by the listening socket on the server machine.
	UseBroker (see page 1787)	Indicates whether an OLEnterprise connection should use the OLEnterprise Business Object Broker.

36.1.1.2 TMidasConnection.Create Constructor

Creates and initializes an instance of TMidasConnection (see page 1782).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TMidasConnection(TComponent * AOwner);
```

Description

Call Create to instantiate a TMidasConnection (see page 1782) component at runtime. TMidasConnection (see page 1782) components placed in forms or data modules at design time are created automatically.

Create initializes the value of the ServerPort (see page 1786) property to 211. This is also the default port number for ScktSrvr.exe and ScktSrvc.exe.

See Also

ServerPort (see page 1786)

Create (see page 1291)

Port

36.1.1.3 TMidasConnection Methods

The methods of the TMidasConnection class are listed here.

Protected Methods

	Name	Description
	DoConnect (see page 1784)	Connects to an application server.
	DoDisconnect (see page 1785)	Drops a connection to an application server.
	SetConnectType (see page 1785)	This is SetConnectType, a member of class TMidasConnection.
	SetUseBroker (see page 1785)	This is SetUseBroker, a member of class TMidasConnection.

36.1.1.3.1 TMidasConnection.DoConnect Method

Connects to an application server.

Pascal

```
procedure DoConnect; override;
```

C++

```
virtual __fastcall DoConnect();
```

Description

DoConnect is called when the Connected property is set to true. DoConnect establishes a dual interface with the remote data module on the application server. This interface is used to locate providers on the application server.

To locate the server system, DoConnect uses the ComputerName property. If the ObjectBroker property is set, DoConnect gets this value from the object broker. Otherwise, the application must assign this property a value directly.

See Also

AfterConnect

BeforeConnect

Connected (see page 1296)

AppServer

ComputerName (see page 1287)

ObjectBroker (see page 1300)

36.1.1.3.2 TMidasConnection.DoDisconnect Method

Drops a connection to an application server.

Pascal

```
procedure DoDisconnect; override;
```

C++

```
virtual __fastcall DoDisconnect();
```

Description

DoDisconnect is called internally when the Connected property is set to false. It drops the connection and changes the AppServer property to a NULL Variant.

See Also

AfterDisconnect

BeforeDisconnect

Connected (see page 1296)

AppServer

36.1.1.3.3 TMidasConnection.SetConnectType Method

Pascal

```
procedure SetConnectType(Value: TConnectType); virtual;
```

C++

```
virtual __fastcall SetConnectType(TConnectType Value);
```

Description

This is SetConnectType, a member of class TMidasConnection.

36.1.1.3.4 TMidasConnection.SetUseBroker Method

Pascal

```
procedure SetUseBroker(Value: Boolean); virtual;
```

C++

```
virtual __fastcall SetUseBroker(Boolean Value);
```

Description

This is SetUseBroker, a member of class TMidasConnection.

36.1.1.4 TMidasConnection Properties

The properties of the TMidasConnection class are listed here.

Published Properties

	Name	Description
	ConnectType (see page 1786)	Specifies the protocol to use to establish the connection to the application server.
	ServerPort (see page 1786)	Specifies the port used by the listening socket on the server machine.
	UseBroker (see page 1787)	Indicates whether an OLEnterprise connection should use the OLEnterprise Business Object Broker.

36.1.1.4.1 TMidasConnection.ConnectType Property

Specifies the protocol to use to establish the connection to the application server.

Pascal

```
property ConnectType: TConnectType;
```

C++

```
__property TConnectType ConnectType;
```

Description

Use ConnectType to specify the desired protocol for connecting with the application server.

See Also

ServerPort (see page 1786)

ComputerName (see page 1287)

UseBroker (see page 1787)

ServerName (see page 1301)

36.1.1.4.2 TMidasConnection.ServerPort Property

Specifies the port used by the listening socket on the server machine.

Pascal

```
property ServerPort: Integer;
```

C++

```
__property int ServerPort;
```

Description

When establishing connections to the application server using TCP/IP sockets, the application server must have a socket dispatcher listening on a predefined port for requests from client applications. If the value of ConnectType (see page 1786) is ctSockets, set ServerPort to the port number used by Scktsrvr.exe or SckTSrvc.exe on the server system. TMidasConnection (see page 1782) attempts to connect to the application server by sending a message to the system specified by the ComputerName property at the port specified by ServerPort.

Note: ServerPort has no effect if ConnectType (see page 1786) is not ctSockets.

See Also

ConnectType (see page 1786)

ComputerName (see page 1287)

Port

36.1.1.4.3 TMidasConnection.UseBroker Property

Indicates whether an OLEnterprise connection should use the OLEnterprise Business Object Broker.

Pascal

```
property UseBroker: Boolean;
```

C++

```
__property Boolean UseBroker;
```

Description

When establishing connections to the application server using OLEnterprise, UseBroker specifies whether the connection should use the Business Object Broker. If UseBroker is true, TMidasConnection (see page 1782) uses the Business Object Broker, and the ComputerName property specifies the name of the Business Object Broker. If UseBroker is false, TMidasConnection (see page 1782) uses OLEnterprise, but does not go through the Business Object Broker. When UseBroker is false, ComputerName specifies the name of the OLEnterprise server.

For more information about using OLEnterprise, see the OLEnterprise documentation.

Note: The value of UseBroker has no effect unless ConnectType (see page 1786) is ctOLEnterprise.

See Also

ConnectType (see page 1786)

ComputerName (see page 1287)

36.1.2 TRemoteServer Class

TRemoteServer handles a DCOM connection to a remote server in a client application that is part of a multi-tiered application.

Class Hierarchy

Pascal

```
TRemoteServer = class(TDCOMConnection);
```

C++

```
class TRemoteServer : public TDCOMConnection;
```

File

MidasCon

Description

TRemoteServer is provided for backward compatibility. When writing client applications that connect to application servers using DCOM, use TDCOMConnection (see page 1284) instead.

See Also

TClientDataSet

TDCOMConnection (see page 1284)

TSocketConnection

TWebConnection

TDataSetProvider (see page 3145)

36.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TConnectType (see page 1789)	This is record MidasCon.TConnectType.

36.2.1 MidasCon.TConnectType Enumeration

Pascal

```
TConnectType = (
  ctDCOM,
  ctSockets,
  ctOLEnterprise
);
```

C++

```
enum TConnectType {
  ctDCOM,
  ctSockets,
  ctOLEnterprise
};
```

File

MidasCon

Description

This is record MidasCon.TConnectType.

37 MidItems Namespace

This is namespace MidItems.

Topics

Name	Description
Classes (see page 1796)	The following table lists classes in this documentation.
Interfaces (see page 2200)	The following table lists interfaces in this documentation.
Functions (see page 2232)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 2237)	The following table lists structs, records, enums in this documentation.
Types (see page 2240)	The following table lists types in this documentation.
Constants (see page 2242)	The following table lists constants in this documentation.

Classes

	Name	Description
	TApplyUpdatesButton (see page 1798)	TApplyUpdatesButton generates an HTML control that sends an Update message to the XML broker on an InternetExpress application.
	TCaptionAttributes (see page 1804)	TCaptionAttributes customizes the appearance of the label that identifies a display control in a WebSnap or InternetExpress application.
	TCustomDataForm (see page 1808)	TCustomDataForm is the base class for components that generate an HTML form in an InternetExpress application.
	TCustomDataGrid (see page 1811)	TCustomDataGrid is the base class for Web items that generate an HTML table to display a dataset in an XML data packet.
	TCustomDataNavigator (see page 1819)	TCustomDataNavigator assembles the HTML generated by a set of TDataSetButton (see page 1859) and TXMLButton (see page 2151) descendants.
	TCustomFieldGroup (see page 1824)	TCustomFieldGroup is the base class for Web items that arrange information on an HTML form as a set of labeled fields.
	TCustomLayoutGroup (see page 1830)	TCustomLayoutGroup is the base class for Web items that arrange content produced by arbitrary Web items.
	TCustomQueryButtons (see page 1836)	TCustomQueryButtons assembles the HTML generated by a set of TQueryButton (see page 1953) descendants.
	TCustomQueryForm (see page 1841)	TCustomQueryForm is the base class for Web items that generate HTML forms for letting users enter application-defined values.
	TDataForm (see page 1847)	TDataForm generates an HTML form that displays database information and lets users apply updates.
	TDataGrid (see page 1850)	TDataGrid generates an HTML table to display a dataset in an XML data packet.
	TDataNavigator (see page 1855)	TDataNavigator assembles the HTML generated by a set of TDataSetButton (see page 1859) and TXMLButton (see page 2151) descendants.
	TDataSetButton (see page 1859)	TDataSetButton is the base class for Web items that create HTML buttons for navigating or editing data.

	TDeleteButton (see page 1867)	TDeleteButton generates an HTML control that deletes a record from an XML data packet.
	TFieldGroup (see page 1869)	TFieldGroup arranges database information on an HTML form as a set of labeled fields.
	TFieldRadioGroup (see page 1874)	TFieldRadioGroup creates an HTML radio group control to represent a field in an XML data packet.
	TFieldSelectOptions (see page 1884)	TFieldSelectOptions creates an HTML list control to represent a field in an XML data packet.
	TFieldStatus (see page 1894)	TFieldStatus generates an HTML control that indicates whether a record in an XML data packet has been modified by the end user.
	TFieldText (see page 1899)	TFieldText creates a single-line HTML input control to represent a field in an XML data packet.
	TFieldTextArea (see page 1905)	TFieldTextArea creates a multi-line HTML input control to represent a field in an XML data packet.
	TFirstButton (see page 1913)	TFirstButton generates an HTML control that moves to the first record in an XML data packet.
	TFormLayout (see page 1915)	TFormLayout lets child Web items adhere to layout rules imposed by their parent.
	TGridAttributes (see page 1923)	TGridAttributes describes the options the TDataGrid (see page 1850) component adds to an HTML <TABLE> tag.
	TGridRowAttributes (see page 1929)	TGridRowAttributes describes the options the TDataGrid (see page 1850) component adds to its HTML row tags (<TR>).
	TInsertButton (see page 1935)	TInsertButton generates an HTML control that inserts a new record in an XML data packet.
	TLastButton (see page 1937)	TLastButton generates an HTML control that moves to the last record in an XML data packet.
	TLayoutGroup (see page 1939)	TLayoutGroup generates the HTML to arrange content produced by other Web items.
	TNextButton (see page 1943)	TNextButton generates an HTML control that moves to the next record in an XML data packet.
	TNextPageButton (see page 1945)	TNextPageButton generates an HTML control that moves to the next set of records in an XML data packet.
	TPostButton (see page 1947)	TPostButton generates an HTML control that posts changes to an XML delta packet.
	TPriorButton (see page 1949)	TPriorButton generates an HTML control that moves to the previous record in an XML data packet.
	TPriorPageButton (see page 1951)	TPriorPageButton generates an HTML control that moves to the previous set of records in an XML data packet.
	TQueryButton (see page 1953)	TQueryButton is the base class for Web items that create HTML buttons that manage application-defined values.
	TQueryButtons (see page 1962)	TQueryButtons assembles the HTML generated by a set of TQueryButton (see page 1953) descendants.
	TQueryFieldGroup (see page 1965)	TQueryFieldGroup arranges information on an HTML form as a set of labeled fields.
	TQueryForm (see page 1971)	TQueryForm generates HTML forms for letting users enter application-defined values.
	TQueryRadioGroup (see page 1976)	TQueryRadioGroup creates an HTML radio group control to represent named parameter.
	TQuerySelectOptions (see page 1989)	TQuerySelectOptions creates an HTML list control to represent a named parameter.
	TQueryText (see page 2000)	TQueryText creates a single-line HTML input control to represent a named parameter.
	TQueryTextArea (see page 2010)	TQueryTextArea creates a multi-line HTML input control to represent named parameter.
	TResetQueryButton (see page 2022)	TResetQueryButton generates an HTML control that backs out changes to the values on a query form.
	TSelectOptionsColumn (see page 2024)	TSelectOptionsColumn creates an HTML list control to represent a field value in a data grid.
	TStatusColumn (see page 2033)	TStatusColumn generates an HTML control that indicates whether a record in a data grid has been modified by the end user.
	TSubmitQueryButton (see page 2038)	TSubmitQueryButton generates an HTML control that backs out changes to the values on a query form.
	TTextAreaColumn (see page 2041)	TTextAreaColumn creates a multi-line HTML input control to represent a field value in a data grid.

	TTextColumn (see page 2049)	TTextColumn creates a single-line HTML input control to represent a field value in a data grid.
	TUndoButton (see page 2055)	TUndoButton generates an HTML control that backs out changes to the current record.
	TWebButton (see page 2058)	TWebButton is the base class for Web items that create HTML button controls.
	TWebControlGroup (see page 2064)	TWebControlGroup is the base class for Web items that arrange HTML controls.
	TWebDataDisplay (see page 2073)	TWebDataDisplay is the base class for Web items that create HTML display controls.
	TWebDataInput (see page 2087)	TWebDataInput is the base class for Web items that create HTML display controls for entering information.
	TWebDisplay (see page 2095)	This is class MidItems.TWebDisplay.
	TWebDisplayReferenceGroup (see page 2098)	TWebDisplayReferenceGroup is the base class for Web items that arrange a set of controls that influence a control that displays data.
	TWebForm (see page 2102)	TWebForm is the base class for Web items that create HTML forms in InternetExpress applications.
	TWebListInput (see page 2113)	TWebListInput is the base class for Web items that create HTML display controls which present a fixed set of possible values.
	TWebRadioGroupInput (see page 2122)	TWebRadioGroupInput is the base class for Web items that create HTML radio group controls.
	TWebSelectOptionsInput (see page 2128)	TWebSelectOptionsInput is the base class for Web items that create HTML list controls.
	TWebStatus (see page 2133)	TWebStatus is the base class for Web items that indicate the status of records in an XML data packet.
	TWebTextAreaInput (see page 2138)	TWebTextAreaInput is the base class for Web items that create HTML multi-line text controls.
	TWebTextInput (see page 2145)	TWebTextInput is the base class for Web items that create HTML single-line text controls.
	TXMLButton (see page 2151)	TXMLButton is the base class for Web items that create HTML buttons that perform some action with the XML data packet.
	TXMLData (see page 2157)	TXMLData manages the relationship between an XML broker and a Web item that represents a dataset in that XML broker's data packet.
	TXMLDataParent (see page 2161)	This is class MidItems.TXMLDataParent.
	TXMLDataSet (see page 2164)	TXMLDataSet manages the relationship between a Web item and the XML dataset to which it is bound.
	TXMLDataSetParent (see page 2172)	This is class MidItems.TXMLDataSetParent.
	TXMLDisplayGroup (see page 2176)	TXMLDisplayGroup is the base class for Web items that arrange a set of special-purpose controls in an HTML form.
	TXMLDisplayParent (see page 2188)	This is class MidItems.TXMLDisplayParent.
	TXMLDisplayReferenceButton (see page 2191)	TXMLDisplayReferenceButton is the base class for Web items that generate HTML buttons for influencing controls that display data.
	TXMLDisplayReferenceGroup (see page 2197)	TXMLDisplayReferenceGroup is the base class for Web items that arrange a set of button controls that influence a control that displays data.

Constants

Name	Description
FormMethodNames (see page 2242)	This is constant MidItems.FormMethodNames.
HTMLAlignValues (see page 2243)	This is constant MidItems.HTMLAlignValues.
HTMLVAlignValues (see page 2243)	This is constant MidItems.HTMLVAlignValues.
sActionAttr (see page 2243)	This is constant MidItems.sActionAttr.
sAlignAttr (see page 2244)	This is constant MidItems.sAlignAttr.
sBgColorAttr (see page 2244)	This is constant MidItems.sBgColorAttr.
sBorderAttr (see page 2244)	This is constant MidItems.sBorderAttr.
sCellPaddingAttr (see page 2245)	This is constant MidItems.sCellPaddingAttr.
sCellSpacingAttr (see page 2245)	This is constant MidItems.sCellSpacingAttr.
sCheckedAttr (see page 2245)	This is constant MidItems.sCheckedAttr.
sClassAttr (see page 2246)	This is constant MidItems.sClassAttr.
sColsAttr (see page 2246)	This is constant MidItems.sColsAttr.
sColSpanAttr (see page 2246)	This is constant MidItems.sColSpanAttr.
sEncTypeAttr (see page 2247)	This is constant MidItems.sEncTypeAttr.
sImageHeightAttr (see page 2247)	This is constant MidItems.sImageHeightAttr.

sImageWidthAttr (see page 2247)	This is constant MidItems.sImageWidthAttr.
sMaxLengthAttr (see page 2248)	This is constant MidItems.sMaxLengthAttr.
sMethodAttr (see page 2248)	This is constant MidItems.sMethodAttr.
sMultipleAttr (see page 2248)	This is constant MidItems.sMultipleAttr.
sNameAttr (see page 2249)	This is constant MidItems.sNameAttr.
sOnClickAttr (see page 2249)	This is constant MidItems.sOnClickAttr.
sReadOnlyAttr (see page 2249)	This is constant MidItems.sReadOnlyAttr.
sRowsAttr (see page 2250)	This is constant MidItems.sRowsAttr.
sSelectedAttr (see page 2250)	This is constant MidItems.sSelectedAttr.
sSizeAttr (see page 2250)	This is constant MidItems.sSizeAttr.
sStyleAttr (see page 2251)	This is constant MidItems.sStyleAttr.
sTabIndexAttr (see page 2251)	This is constant MidItems.sTabIndexAttr.
sTypeAttr (see page 2251)	This is constant MidItems.sTypeAttr.
sVAlignAttr (see page 2252)	This is constant MidItems.sVAlignAttr.
sValueAttr (see page 2252)	This is constant MidItems.sValueAttr.
sWidthAttr (see page 2252)	This is constant MidItems.sWidthAttr.
sWrapAttr (see page 2253)	This is constant MidItems.sWrapAttr.
TextAreaWrap (see page 2253)	This is constant MidItems.TextAreaWrap.

Functions

	Name	Description
	AddBoolAttrib (see page 2232)	This is function MidItems.AddBoolAttrib.
	AddCustomAttrib (see page 2232)	This is function MidItems.AddCustomAttrib.
	AddIntAttrib (see page 2233)	This is function MidItems.AddIntAttrib.
	AddQuotedAttrib (see page 2233)	This is function MidItems.AddQuotedAttrib.
	AddStringAttrib (see page 2233)	This is function MidItems.AddStringAttrib.
	CanAddClassHelper (see page 2234)	This is function MidItems.CanAddClassHelper.
	CreateDefaultButtonClasses (see page 2234)	This is function MidItems.CreateDefaultButtonClasses.
	DeclareSubmitForm (see page 2234)	This is function MidItems.DeclareSubmitForm.
	FindDispatcher (see page 2235)	This is function MidItems.FindDispatcher.
	FindProducer (see page 2235)	This is function MidItems.FindProducer.
	FormatColumnData (see page 2235)	This is function MidItems.FormatColumnData.
	FormatColumnHeading (see page 2236)	This is function MidItems.FormatColumnHeading.
	GetItemValuesFromDataSet (see page 2236)	This is function MidItems.GetItemValuesFromDataSet.

Interfaces

	Name	Description
	IDataSetComponent (see page 2200)	IDataSetComponent defines the methods that bind a Web item to an XML broker.
	IDataSetField (see page 2204)	This is class MidItems.IDataSetField.
	IDataSetFields (see page 2206)	This is class MidItems.IDataSetFields.
	IFormatColumn (see page 2208)	This is class MidItems.IFormatColumn.
	IHTMLField (see page 2209)	IHTMLField is an interface that must be implemented by Web items that generate <INPUT> elements.
	IHTMLForm (see page 2211)	IHTMLForm is an interface that must be implemented by Web items that generate forms.
	IInetXWebDataSet (see page 2213)	IInetXWebDataSet lets a Web item interact with information about a dataset in an XML broker's data packet.
	IQueryField (see page 2217)	This is class MidItems.IQueryField.
	IQueryFields (see page 2220)	This is class MidItems.IQueryFields.
	IQueryForm (see page 2221)	This is class MidItems.IQueryForm.
	IRestoreDefaults (see page 2221)	IRestoreDefaults enables the Web Page editor to restore a Web item to a default state.
	IStatusField (see page 2223)	This is class MidItems.IStatusField.
	IValidateField (see page 2223)	This is class MidItems.IValidateField.
	IValidateFields (see page 2224)	This is class MidItems.IValidateFields.
	IWebDisplay (see page 2226)	

	IWebDisplayReference (see page 2227)	
	IXMLDisplay (see page 2229)	This is class MidItems.IXMLDisplay.

Structs, Records, Enums

	Name	Description
	TCaptionPosition (see page 2237)	TCaptionPosition indicates the relative position of a label to the HTML control it labels.
	TFormMethod (see page 2238)	TFormMethod represents the method attribute on the <form> tag of an HTML form.
	TLayoutState (see page 2238)	This is record MidItems.TLayoutState.
	TTextAreaWrap (see page 2239)	TTextAreaWrap indicates how an HTML display control should treat items that are too wide for the physical width of the control.

Types

Name	Description
IMidasWebDataSet (see page 2240)	This is type MidItems.IMidasWebDataSet.
TWebButtonClass (see page 2240)	This is type MidItems.TWebButtonClass.
TWebDataDisplayClass (see page 2241)	This is type MidItems.TWebDataDisplayClass.
TWebDataInputClass (see page 2241)	This is type MidItems.TWebDataInputClass.
TWebStatusClass (see page 2241)	This is type MidItems.TWebStatusClass.

37.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TApplyUpdatesButton (see page 1798)	TApplyUpdatesButton generates an HTML control that sends an Update message to the XML broker on an InternetExpress application.
	TCaptionAttributes (see page 1804)	TCaptionAttributes customizes the appearance of the label that identifies a display control in a WebSnap or InternetExpress application.
	TCustomDataForm (see page 1808)	TCustomDataForm is the base class for components that generate an HTML form in an InternetExpress application.
	TCustomDataGrid (see page 1811)	TCustomDataGrid is the base class for Web items that generate an HTML table to display a dataset in an XML data packet.
	TCustomDataNavigator (see page 1819)	TCustomDataNavigator assembles the HTML generated by a set of TDataSetButton (see page 1859) and TXMLButton (see page 2151) descendants.
	TCustomFieldGroup (see page 1824)	TCustomFieldGroup is the base class for Web items that arrange information on an HTML form as a set of labeled fields.
	TCustomLayoutGroup (see page 1830)	TCustomLayoutGroup is the base class for Web items that arrange content produced by arbitrary Web items.
	TCustomQueryButtons (see page 1836)	TCustomQueryButtons assembles the HTML generated by a set of TQueryButton (see page 1953) descendants.
	TCustomQueryForm (see page 1841)	TCustomQueryForm is the base class for Web items that generate HTML forms for letting users enter application-defined values.
	TDataForm (see page 1847)	TDataForm generates an HTML form that displays database information and lets users apply updates.
	TDataGrid (see page 1850)	TDataGrid generates an HTML table to display a dataset in an XML data packet.
	TDataNavigator (see page 1855)	TDataNavigator assembles the HTML generated by a set of TDataSetButton (see page 1859) and TXMLButton (see page 2151) descendants.
	TDataSetButton (see page 1859)	TDataSetButton is the base class for Web items that create HTML buttons for navigating or editing data.
	TDeleteButton (see page 1867)	TDeleteButton generates an HTML control that deletes a record from an XML data packet.
	TFieldGroup (see page 1869)	TFieldGroup arranges database information on an HTML form as a set of labeled fields.
	TFieldRadioGroup (see page 1874)	TFieldRadioGroup creates an HTML radio group control to represent a field in an XML data packet.
	TFieldSelectOptions (see page 1884)	TFieldSelectOptions creates an HTML list control to represent a field in an XML data packet.
	TFieldStatus (see page 1894)	TFieldStatus generates an HTML control that indicates whether a record in an XML data packet has been modified by the end user.
	TFieldText (see page 1899)	TFieldText creates a single-line HTML input control to represent a field in an XML data packet.
	TFieldTextArea (see page 1905)	TFieldTextArea creates a multi-line HTML input control to represent a field in an XML data packet.
	TFirstButton (see page 1913)	TFirstButton generates an HTML control that moves to the first record in an XML data packet.
	TFormLayout (see page 1915)	TFormLayout lets child Web items adhere to layout rules imposed by their parent.
	TGridAttributes (see page 1923)	TGridAttributes describes the options the TDataGrid (see page 1850) component adds to an HTML <TABLE> tag.
	TGridRowAttributes (see page 1929)	TGridRowAttributes describes the options the TDataGrid (see page 1850) component adds to its HTML row tags (<TR>).
	TInsertButton (see page 1935)	TInsertButton generates an HTML control that inserts a new record in an XML data packet.
	TLastButton (see page 1937)	TLastButton generates an HTML control that moves to the last record in an XML data packet.

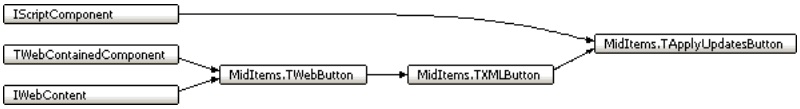
	TLayoutGroup (see page 1939)	TLayoutGroup generates the HTML to arrange content produced by other Web items.
	TNextButton (see page 1943)	TNextButton generates an HTML control that moves to the next record in an XML data packet.
	TNextPageButton (see page 1945)	TNextPageButton generates an HTML control that moves to the next set of records in an XML data packet.
	TPostButton (see page 1947)	TPostButton generates an HTML control that posts changes to an XML delta packet.
	TPriorButton (see page 1949)	TPriorButton generates an HTML control that moves to the previous record in an XML data packet.
	TPriorPageButton (see page 1951)	TPriorPageButton generates an HTML control that moves to the previous set of records in an XML data packet.
	TQueryButton (see page 1953)	TQueryButton is the base class for Web items that create HTML buttons that manage application-defined values.
	TQueryButtons (see page 1962)	TQueryButtons assembles the HTML generated by a set of TQueryButton (see page 1953) descendants.
	TQueryFieldGroup (see page 1965)	TQueryFieldGroup arranges information on an HTML form as a set of labeled fields.
	TQueryForm (see page 1971)	TQueryForm generates HTML forms for letting users enter application-defined values.
	TQueryRadioGroup (see page 1976)	TQueryRadioGroup creates an HTML radio group control to represent named parameter.
	TQuerySelectOptions (see page 1989)	TQuerySelectOptions creates an HTML list control to represent a named parameter.
	TQueryText (see page 2000)	TQueryText creates a single-line HTML input control to represent a named parameter.
	TQueryTextArea (see page 2010)	TQueryTextArea creates a multi-line HTML input control to represent named parameter.
	TResetQueryButton (see page 2022)	TResetQueryButton generates an HTML control that backs out changes to the values on a query form.
	TSelectOptionsColumn (see page 2024)	TSelectOptionsColumn creates an HTML list control to represent a field value in a data grid.
	TStatusColumn (see page 2033)	TStatusColumn generates an HTML control that indicates whether a record in a data grid has been modified by the end user.
	TSubmitQueryButton (see page 2038)	TSubmitQueryButton generates an HTML control that backs out changes to the values on a query form.
	TTextAreaColumn (see page 2041)	TTextAreaColumn creates a multi-line HTML input control to represent a field value in a data grid.
	TTextColumn (see page 2049)	TTextColumn creates a single-line HTML input control to represent a field value in a data grid.
	TUndoButton (see page 2055)	TUndoButton generates an HTML control that backs out changes to the current record.
	TWebButton (see page 2058)	TWebButton is the base class for Web items that create HTML button controls.
	TWebControlGroup (see page 2064)	TWebControlGroup is the base class for Web items that arrange HTML controls.
	TWebDataDisplay (see page 2073)	TWebDataDisplay is the base class for Web items that create HTML display controls.
	TWebDataInput (see page 2087)	TWebDataInput is the base class for Web items that create HTML display controls for entering information.
	TWebDisplay (see page 2095)	This is class MidItems.TWebDisplay.
	TWebDisplayReferenceGroup (see page 2098)	TWebDisplayReferenceGroup is the base class for Web items that arrange a set of controls that influence a control that displays data.
	TWebForm (see page 2102)	TWebForm is the base class for Web items that create HTML forms in InternetExpress applications.
	TWebListInput (see page 2113)	TWebListInput is the base class for Web items that create HTML display controls which present a fixed set of possible values.
	TWebRadioGroupInput (see page 2122)	TWebRadioGroupInput is the base class for Web items that create HTML radio group controls.
	TWebSelectOptionsInput (see page 2128)	TWebSelectOptionsInput is the base class for Web items that create HTML list controls.
	TWebStatus (see page 2133)	TWebStatus is the base class for Web items that indicate the status of records in an XML data packet.
	TWebTextAreaInput (see page 2138)	TWebTextAreaInput is the base class for Web items that create HTML multi-line text controls.

	TWebTextInput (see page 2145)	TWebTextInput is the base class for Web items that create HTML single-line text controls.
	TXMLButton (see page 2151)	TXMLButton is the base class for Web items that create HTML buttons that perform some action with the XML data packet.
	TXMLData (see page 2157)	TXMLData manages the relationship between an XML broker and a Web item that represents a dataset in that XML broker's data packet.
	TXMLDataParent (see page 2161)	This is class MidItems.TXMLDataParent.
	TXMLDataSet (see page 2164)	TXMLDataSet manages the relationship between a Web item and the XML dataset to which it is bound.
	TXMLDataSetParent (see page 2172)	This is class MidItems.TXMLDataSetParent.
	TXMLDisplayGroup (see page 2176)	TXMLDisplayGroup is the base class for Web items that arrange a set of special-purpose controls in an HTML form.
	TXMLDisplayParent (see page 2188)	This is class MidItems.TXMLDisplayParent.
	TXMLDisplayReferenceButton (see page 2191)	TXMLDisplayReferenceButton is the base class for Web items that generate HTML buttons for influencing controls that display data.
	TXMLDisplayReferenceGroup (see page 2197)	TXMLDisplayReferenceGroup is the base class for Web items that arrange a set of button controls that influence a control that displays data.

37.1.1 TApplyUpdatesButton Class

TApplyUpdatesButton generates an HTML control that sends an Update message to the XML broker on an InternetExpress application.

Class Hierarchy



Pascal

```
TApplyUpdatesButton = class(TXMLButton, IScriptComponent);
```

C++

```
class TApplyUpdatesButton : public TXMLButton, public IScriptComponent;
```

File

MidItems

Description

Add TApplyUpdatesButton as the child of a TDataNavigator (see page 1855) component (or a layout group that is a child of a TDataNavigator (see page 1855) component) to enable the generated HTML form to submit changes made by end users. The resulting HTML button's action generates an HTTP message that forwards all changes to a specific XML data packet back to the XML broker on the InternetExpress application.

Use the properties of TApplyUpdatesButton to customize the appearance of the button and associate it with an XML broker. The associated XML broker adds the XML data packet to the HTML document and responds to update messages generated by the apply updates button on the HTML form.

See Also

TXMLBroker

TDataNavigator (see page 1855)

TLayoutGroup (see page 1939)

Using the Web Page Editor

37.1.1.1 TApplyUpdatesButton Members

The following tables list the members exposed by TApplyUpdatesButton.

Protected Methods

	Name	Description
	AddElements (see page 1800)	This is AddElements, a member of class TApplyUpdatesButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
V	GetCaption (see page 2154)	Returns the value of the Caption property.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
	GetSubComponents (see page 1800)	This is GetSubComponents, a member of class TApplyUpdatesButton.
	GetXMLBroker (see page 2154)	This is GetXMLBroker, a member of class TXMLButton.
	GetXMLUseParent (see page 2154)	This is GetXMLUseParent, a member of class TXMLButton.
V	ImplAddElements (see page 1800)	This is ImplAddElements, a member of class TApplyUpdatesButton.
V	ImplContent (see page 1800)	Provides the underlying implementation for the Content method.
V	Notification (see page 2154)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetXMLBroker (see page 2155)	This is SetXMLBroker, a member of class TXMLButton.
	SetXMLUseParent (see page 2155)	This is SetXMLUseParent, a member of class TXMLButton.

Protected Properties

	Name	Description
	DefaultCaption (see page 2156)	This is DefaultCaption, a member of class TXMLButton.
R	XMLData (see page 2156)	This is XMLData, a member of class TXMLButton.

Public Methods

	Name	Description
V	Create (see page 1799)	Creates and initializes an instance of TApplyUpdatesButton (see page 1798).
V	Destroy (see page 2153)	Frees an instance of TXMLButton (see page 2151).

Published Properties

	Name	Description
	Caption (see page 1801)	Specifies the caption that appears on the button.
	Custom (see page 1802)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1802)	Adds style attributes to the generated button control.
	StyleRule (see page 1802)	Applies a style in the page producer's style sheet to the button.
	XMLBroker (see page 1803)	Associates the XML button with an XML broker.
	XMLUseParent (see page 1803)	Specifies whether the XML button is associated with its parent's XML broker.

37.1.1.2 TApplyUpdatesButton.Create Constructor

Creates and initializes an instance of TApplyUpdatesButton (see page 1798).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TApplyUpdatesButton(TComponent * AOwner);
```

Description

Call Create to instantiate TApplyUpdatesButton (see page 1798) at runtime. Buttons added at design time using the Web

Page editor are instantiated automatically.

37.1.1.3 TApplyUpdatesButton Methods

The methods of the TApplyUpdatesButton class are listed here.

Protected Methods

	Name	Description
	AddElements (see page 1800)	This is AddElements, a member of class TApplyUpdatesButton.
	GetSubComponents (see page 1800)	This is GetSubComponents, a member of class TApplyUpdatesButton.
V	ImplAddElements (see page 1800)	This is ImplAddElements, a member of class TApplyUpdatesButton.
V	ImplContent (see page 1800)	Provides the underlying implementation for the Content method.

37.1.1.3.1 TApplyUpdatesButton.AddElements Method

Pascal

```
procedure AddElements(AddIntf: IAddScriptElements);
```

C++

```
__fastcall AddElements(IAddScriptElements AddIntf);
```

Description

This is AddElements, a member of class TApplyUpdatesButton.

37.1.1.3.2 TApplyUpdatesButton.GetSubComponents Method

Pascal

```
function GetSubComponents: TObject;
```

C++

```
__fastcall TObject * GetSubComponents();
```

Description

This is GetSubComponents, a member of class TApplyUpdatesButton.

37.1.1.3.3 TApplyUpdatesButton.ImplAddElements Method

Pascal

```
procedure ImplAddElements(AddIntf: IAddScriptElements); virtual;
```

C++

```
virtual __fastcall ImplAddElements(IAddScriptElements AddIntf);
```

Description

This is ImplAddElements, a member of class TApplyUpdatesButton.

37.1.1.3.4 TApplyUpdatesButton.ImplContent Method

Provides the underlying implementation for the Content method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content method calls ImplContent to generate the HTML that defines the button control. TWebButton introduces ImplContent as an abstract or, in C++ terminology, pure virtual method. TWebButton descendants must override ImplContent to generate the definition of their particular button control.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the Web button should format the HTML control it generates. It implements the ILayoutWebContent interface.

ImplContent returns a string of HTML defining the button control.

See Also

Content (see page 2060)

ILayoutWebContent

37.1.1.4 TApplyUpdatesButton Properties

The properties of the TApplyUpdatesButton class are listed here.

Published Properties

	Name	Description
	Caption (see page 1801)	Specifies the caption that appears on the button.
	Custom (see page 1802)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1802)	Adds style attributes to the generated button control.
	StyleRule (see page 1802)	Applies a style in the page producer's style sheet to the button.
	XMLBroker (see page 1803)	Associates the XML button with an XML broker.
	XMLUseParent (see page 1803)	Specifies whether the XML button is associated with its parent's XML broker.

37.1.1.4.1 TApplyUpdatesButton.Caption Property

Specifies the caption that appears on the button.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the caption that appears on the generated HTML button. The value of Caption is used as the value of a VALUE parameter on the HTML INPUT tag generated to implement the button.

See Also

Custom (see page 2062)

GetCaption (see page 2060)

SetCaption (see page 2061)

37.1.1.4.2 TApplyUpdatesButton.Custom Property

Represents a tag option and its value in the generated INPUT tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

Use Custom to represent one or more options that qualify the INPUT tag generated by the Web button. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to add an ALIGN option that aligns the caption and a TABINDEX tag to control tab order, set Custom to

```
ALIGN=TOP TABINDEX=2
```

See Also

Caption (see page 2062)

Style (see page 2063)

37.1.1.4.3 TApplyUpdatesButton.Style Property

Adds style attributes to the generated button control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to add a STYLE option that qualifies the INPUT tag generated by the Web button. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated INPUT tag as the value of a STYLE option.

See Also

Custom (see page 2062)

StyleRule (see page 2063)

37.1.1.4.4 TApplyUpdatesButton.StyleRule Property

Applies a style in the page producer's style sheet to the button.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated button control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the button's Style property.

See Also

Style (see page 2063)

Styles (see page 2272)

StylesFile (see page 2273)

Styles

StylesFile

37.1.1.4.5 TApplyUpdatesButton.XMLBroker Property

Associates the XML button with an XML broker.

Pascal

```
property XMLBroker: TXMLBroker;
```

C++

```
__property TXMLBroker XMLBroker;
```

Description

Set XMLBroker to specify the XML broker whose data packets the XML button manipulates.

When XML buttons are used by TDataNavigator, the data navigator already establishes a connection (indirectly) to an XML broker. If the XML button uses this same XML broker, you do not need to set the XMLBroker property. Instead, setting the XMLUseParent property to true (the default) automatically associates the XML button with the same broker as the one associated with the navigator.

See Also

XMLUseParent (see page 2156)

37.1.1.4.6 TApplyUpdatesButton.XMLUseParent Property

Specifies whether the XML button is associated with its parent's XML broker.

Pascal

```
property XMLUseParent: Boolean;
```

C++

```
__property Boolean XMLUseParent;
```

Description

Use XMLUseParent to indicate how the XML button determines the XML broker whose data packets it manipulates. When XMLUseParent is false, the XMLBroker property must be set to indicate the XML broker. When XMLUseParent is true, the XML broker associated with the XML button's parent determines the associated XML broker.

The XML button's parent is the Web item (typically TDataNavigator) that uses the XML button to assemble its own HTML content. The parent's associated broker may be defined through an indirect association. For example, the data navigator acts on another component, such as a field group, which has an associated broker.

If the parent does not have an associated XML broker, setting XMLUseParent to true sets the XMLBroker property to nil (Delphi) or NULL (C++).

See Also

XMLBroker (see page 2156)

37.1.2 TCaptionAttributes Class

TCaptionAttributes customizes the appearance of the label that identifies a display control in a WebSnap or InternetExpress application.

Class Hierarchy

Pascal

```
TCaptionAttributes = class(TPersistent);
```

C++

```
class TCaptionAttributes : public TPersistent;
```

File

MidItems

Description

TCaptionAttributes is the type of the CaptionAttributes property of Web items that generate display controls. It defines several properties that customize the appearance of the label generated by the Web item.

See Also

CaptionAttributes

37.1.2.1 TCaptionAttributes Members

The following tables list the members exposed by TCaptionAttributes.

Public Methods

	Name	Description
	Create (see page 1805)	Creates an instance of TCaptionAttributes (see page 1804).

Public Properties

	Name	Description
R	Parent (see page 1806)	Indicates the Web item that uses TCaptionAttributes (see page 1804) to customize the appearance of its generated caption.

Published Properties

	Name	Description
	Align (see page 1805)	Represents the Align option of an HTML tag.
	Custom (see page 1806)	Represents a tag option and its value.

	Style (see page 1806)	Adds style attributes to the generated caption.
	StyleRule (see page 1807)	Applies a style in the page producer's style sheet to the generated caption.
	VAlign (see page 1807)	Represents the VAlign option of an HTML tag.

37.1.2.2 TCaptionAttributes.Create Constructor

Creates an instance of TCaptionAttributes (see page 1804).

Pascal

```
constructor Create(AParent: TComponent);
```

C++

```
__fastcall TCaptionAttributes(TComponent * AParent);
```

Description

Do not create instances of TCaptionAttributes (see page 1804). This object is instantiated by the Web item that uses it to implement its CaptionAttributes property.

The AParent parameter specifies the Web item that creates a TCaptionAttributes (see page 1804) instance for its CaptionAttributes property. This becomes the value of the Parent (see page 1806) property.

See Also

CaptionAttributes (see page 2084)

Parent (see page 1806)

37.1.2.3 TCaptionAttributes Properties

The properties of the TCaptionAttributes class are listed here.

Public Properties

	Name	Description
R	Parent (see page 1806)	Indicates the Web item that uses TCaptionAttributes (see page 1804) to customize the appearance of its generated caption.

Published Properties

	Name	Description
	Align (see page 1805)	Represents the Align option of an HTML tag.
	Custom (see page 1806)	Represents a tag option and its value.
	Style (see page 1806)	Adds style attributes to the generated caption.
	StyleRule (see page 1807)	Applies a style in the page producer's style sheet to the generated caption.
	VAlign (see page 1807)	Represents the VAlign option of an HTML tag.

37.1.2.3.1 TCaptionAttributes.Align Property

Represents the Align option of an HTML tag.

Pascal

```
property Align: THTMLAlign;
```

C++

```
__property THTMLAlign Align;
```

Description

Use Align to add an Align option to the tag that surrounds the caption. The Align option influences the horizontal alignment of the caption text.

The Web item that generates the caption inserts the specified Align option into the HTML tag when it generates its content.

See Also

VAlign (see page 1807)

37.1.2.3.2 TCaptionAttributes.Custom Property

Represents a tag option and its value.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

Use Custom to add one or more options to the tag that surrounds the caption. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces.

The Web item that generates the caption inserts the Custom string into the HTML tag when it generates its content.

See Also

Style (see page 1806)

StyleRule (see page 1807)

37.1.2.3.3 TCaptionAttributes.Parent Property

Indicates the Web item that uses TCaptionAttributes (see page 1804) to customize the appearance of its generated caption.

Pascal

```
property Parent: TComponent;
```

C++

```
__property TComponent * Parent;
```

Description

Use Parent to access the Web item that uses the TCaptionAttributes (see page 1804) object.

See Also

Create (see page 1805)

37.1.2.3.4 TCaptionAttributes.Style Property

Adds style attributes to the generated caption.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated caption. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the HTML tag that generates the caption as the value of a STYLE option.

See Also

Custom (see page 1806)

StyleRule (see page 1807)

37.1.2.3.5 TCaptionAttributes.StyleRule Property

Applies a style in the page producer's style sheet to the generated caption.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated caption. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style (see page 1806) property instead.

See Also

Style (see page 1806)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

37.1.2.3.6 TCaptionAttributes.VAlign Property

Represents the VAlign option of an HTML tag.

Pascal

```
property VAlign: THTMLVAlign;
```

C++

```
__property THTMLVAlign VAlign;
```

Description

Use VAlign to add an VAlign option to the tag that surrounds the caption. The VAlign option influences the vertical alignment of the caption text.

The Web item that generates the caption inserts the specified VAlign option into the HTML tag when it generates its content.

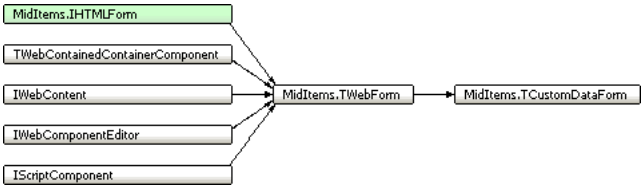
See Also

Align (see page 1805)

37.1.3 TCustomDataForm Class

TCustomDataForm is the base class for components that generate an HTML form in an InternetExpress application.

Class Hierarchy



Pascal

```
TCustomDataForm = class(TWebForm);
```

C++

```
class TCustomDataForm : public TWebForm;
```

File

MidItems

Description

TCustomDataForm is the base class for components that generate HTML forms to display database information that users can view and edit. Such forms can be added to the items of an InternetExpress page producer (InetXPageProducer).

TCustomDataForm implements the ability to generate an HTML form that contains controls for displaying information from an XML data packet. Buttons on the form manipulate that data, and submit updates back to the XML broker that supplies the XML data packet.

TCustomDataForm generates the HTML FORM tag and assembles the HTML generated by its Web items. It restricts the Web items that generate the contents of the form to components that generate data displays (TDataGrid (see page 1850) or TFieldGroup (see page 1869)), components that contain buttons which operate on those displays (TDataNavigator (see page 1855)), and components that manage the layout of HTML controls (TLayoutGroup (see page 1939)). At design time, you can add these Web items to the form using the Web page editor.

See Also

IWebContent

TInetXPageProducer (see page 2285)

37.1.3.1 TCustomDataForm Members

The following tables list the members exposed by TCustomDataForm.

Protected Methods

	Name	Description
	AddElements (see page 2105)	Used internally to generate any javascript declarations needed by the form.
	CanAddClass (see page 2105)	Specifies whether a Web item of a specified type can be added to the WebComponents (see page 2112) property list.
	Content (see page 2106)	Returns the HTML that defines the HTML form.
V	GetDefaultWebComponents (see page 2106)	Returns nil (Delphi) or NULL (C++).
	GetHTMLFormName (see page 2107)	Returns the name of the generated HTML form.
V	GetHTMLFormTag (see page 1810)	This is GetHTMLFormTag, a member of class TCustomDataForm.
	GetHTMLFormVarName (see page 2107)	Returns the name of the javascript variable that refers to the generated form.
V	GetLayoutAttributes (see page 2107)	Returns the object that specifies how the generated form should look.
	GetSubComponents (see page 2108)	Returns the value of the WebComponents (see page 2112) property.
	GetVisibleFields (see page 2108)	Returns the value of the VisibleFields (see page 2112) property.
V	ImplAddElements (see page 2108)	Provides the underlying implementation for the AddElements (see page 2105) method.
V	ImplCanAddClass (see page 1810)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1810)	Provides the underlying implementation for the Content method.
V	SetDefaultWebComponents (see page 2110)	Prevents an application from trying to assign a default set of Web items on the form.

Protected Properties

	Name	Description
R	VisibleFields (see page 2112)	Lists the Web items whose generated content appears in the HTML form.

Public Methods

	Name	Description
V	Create (see page 2104)	Creates and initializes an instance of TWebForm (see page 2102).
V	Destroy (see page 2104)	Frees an instance of TWebForm (see page 2102).
	GetHTMLFormName (see page 2212)	Returns the name of the generated HTML form.
	GetHTMLFormVarName (see page 2212)	Returns the name of the javascript variable that refers to the generated form.

Public Properties

	Name	Description
	Custom (see page 2110)	Represents a tag option and its value in the generated FORM tag.
R	HTMLFormName (see page 2213)	Indicates the name of the generated HTML form.
R	HTMLFormVarName (see page 2213)	Indicates the name of the javascript variable that represents the HTML form.
	Style (see page 2111)	Adds style attributes to the generated form.
	StyleRule (see page 2111)	Applies a style in the page producer's style sheet to the button.
R	WebComponents (see page 2112)	Maintains a list of components that generate HTML content for the TWebForm (see page 2102) descendant.

37.1.3.2 TCustomDataForm Methods

The methods of the TCustomDataForm class are listed here.

Protected Methods

	Name	Description
V	GetHTMLFormTag (see page 1810)	This is GetHTMLFormTag, a member of class TCustomDataForm.
V	ImplCanAddClass (see page 1810)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1810)	Provides the underlying implementation for the Content method.

37.1.3.2.1 TCustomDataForm.GetHTMLFormTag Method

Pascal

```
function GetHTMLFormTag(Options: TWebContentOptions): string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetHTMLFormTag(TWebContentOptions Options);
```

Description

This is GetHTMLFormTag, a member of class TCustomDataForm.

37.1.3.2.2 TCustomDataForm.ImplCanAddClass Method

Provides the underlying implementation for the CanAddClass method.

Pascal

```
function ImplCanAddClass(AParent: TComponent; AClass: TClass): Boolean; override;
```

C++

```
virtual __fastcall Boolean ImplCanAddClass(TComponent * AParent, TClass AClass);
```

Description

The CanAddClass method calls ImpleCanAddClass to determine whether the TWebForm descendant can add a component of type AClass to its WebComponents property list. AParent is the TWebForm instance to which the component will be added.

In TWebForm, ImplCanAddClass is abstract or, in C++ terminology, pure virtual. TWebForm descendants must override ImplCanAddClass to indicate what types of Web Items they can use when generating their HTML forms.

See Also

CanAddClass (see page 2105)

WebComponents (see page 2112)

37.1.3.2.3 TCustomDataForm.ImplContent Method

Provides the underlying implementation for the Content method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content method calls ImplContent to generate the HTML form. ImplContent is anabstract or, in C++ terminology, pure virtual method in TWebForm. TWebForm descendants override it to generate the HTML form.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the TWebForm should format the HTML control it generates. Typically, this parameter is nil (Delphi) or NULL (C++) for forms.

ImplContent returns an HTML form definition.

See Also

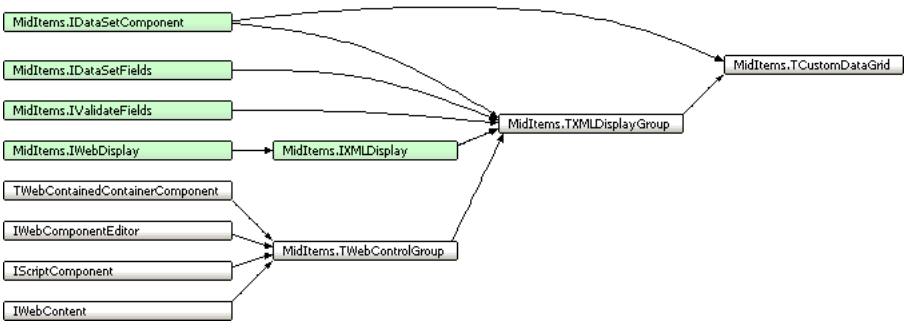
Content (see page 2106)

AddElements (see page 2105)

37.1.4 TCustomDataGrid Class

TCustomDataGrid is the base class for Web items that generate an HTML table to display a dataset in an XML data packet.

Class Hierarchy



Pascal

```
TCustomDataGrid = class(TXMLDisplayGroup, IDataSetComponent);
```

C++

```
class TCustomDataGrid : public TXMLDisplayGroup, public IDataSetComponent;
```

File

MidItems

Description

Use TCustomDataGrid as a base class for Web items that generate an HTML table to display multiple records from a dataset as a set of rows and columns. Each TCustomDataGrid object maintains a list of Web items that generate the individual fields (columns) of the table. These Web items can be from the following classes: TTextColumn (see page 2049), TSelectOptionColumn, TTextAreaColumn (see page 2041), and TStatusColumn (see page 2033).

You can bind the data grid to an XML data packet or dataset field within an XML data packet. The column Web items can then use that binding to determine the source of the data they display.

TCustomDataGrid introduces a number of properties that affect the appearance of the generated table.

In order to let other Web items use the HTML table that the data grid generates, TCustomDataGrid inherits the IWebContent interface from TWebControlGroup (see page 2064).

Do not create instances of TCustomDataGrid. Instead, use a descendant of TCustomDataGrid such as TDataGrid (see page 1850).

See Also

TDataGrid (see page 1850)

IWebContent

TSelectOptionsColumn (see page 2024)

TStatusColumn (see page 2033)

TTextAreaColumn (see page 2041)

TTextColumn (see page 2049)

37.1.4.1 TCustomDataGrid Members

The following tables list the members exposed by TCustomDataGrid.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	CreateDefaultFields (see page 2179)	This is CreateDefaultFields, a member of class TXMLDisplayGroup.
	FindField (see page 2180)	This is FindField, a member of class TXMLDisplayGroup.
	FormatTable (see page 1814)	This is FormatTable, a member of class TCustomDataGrid.
	GetDataSet (see page 2180)	Returns the value of the DataSet property.
	GetDataSetField (see page 2180)	This is GetDataSetField, a member of class TXMLDisplayGroup.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
	GetEnableValidateFields (see page 2180)	This is GetEnableValidateFields, a member of class TXMLDisplayGroup.
V	GetFieldsList (see page 2181)	This is GetFieldsList, a member of class TXMLDisplayGroup.
	GetIsMultipleRecordView (see page 2181)	This is GetIsMultipleRecordView, a member of class TXMLDisplayGroup.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
	GetVisibleFields (see page 2181)	This is GetVisibleFields, a member of class TXMLDisplayGroup.
	GetXMLBroker (see page 2181)	Returns the value of the XMLBroker property.
	GetXMLDataSet (see page 2182)	Returns the value of the XMLDataSet property.
	GetXMLDisplayName (see page 2182)	This is GetXMLDisplayName, a member of class TXMLDisplayGroup.
	GetXMLRowSetName (see page 2182)	This is GetXMLRowSetName, a member of class TXMLDisplayGroup.
	HasStatusField (see page 2182)	This is HasStatusField, a member of class TXMLDisplayGroup.
V	ImplAddElements (see page 1815)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 1815)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1815)	Provides the underlying implementation for the Content method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	ImplGetSubComponents (see page 2183)	Provides the underlying implementation for the GetSubComponents method.
V	ImplIsMultipleRecordView (see page 1816)	This is ImplIsMultipleRecordView, a member of class TCustomDataGrid.
	IsFieldInUse (see page 2184)	This is IsFieldInUse, a member of class TXMLDisplayGroup.
V	Notification (see page 2184)	Forwards notification messages to all owned components.
	SetDataSetField (see page 2184)	This is SetDataSetField, a member of class TXMLDisplayGroup.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.
	SetEnableValidateFields (see page 2185)	This is SetEnableValidateFields, a member of class TXMLDisplayGroup.
	SetHeadingAttributes (see page 1816)	This is SetHeadingAttributes, a member of class TCustomDataGrid.
	SetRowAttributes (see page 1816)	This is SetRowAttributes, a member of class TCustomDataGrid.
	SetTableAttributes (see page 1817)	This is SetTableAttributes, a member of class TCustomDataGrid.
	SetXMLBroker (see page 2185)	This is SetXMLBroker, a member of class TXMLDisplayGroup.
	ValidateDataSet (see page 2185)	This is ValidateDataSet, a member of class TXMLDisplayGroup.
	ValidateFields (see page 2185)	This is ValidateFields, a member of class TXMLDisplayGroup.
V	XMLDataChange (see page 2185)	This is XMLDataChange, a member of class TXMLDisplayGroup.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.
	EnableValidateFields (see page 2186)	This is EnableValidateFields, a member of class TXMLDisplayGroup.
R	XMLDataSet (see page 2187)	Indicates the TXMLDataSet object that represents the dataset to which the Web item is bound.

Public Methods

	Name	Description
V	Create (see page 1814)	Creates and initializes an instance of TCustomDataGrid (see page 1811).
V	Destroy (see page 1814)	Frees an instance of TCustomDataGrid (see page 1811).
	GetDataSet (see page 2201)	Returns the value of the DataSet (see page 2202) property.
	GetEnableValidateFields (see page 2225)	This is GetEnableValidateFields, a member of class IValidateFields.
	GetFieldsList (see page 2207)	This is GetFieldsList, a member of class IDataSetFields.
	GetIsMultipleRecordView (see page 2227)	This is GetIsMultipleRecordView, a member of class IWebDisplay.
	GetIsMultipleRecordView (see page 2229)	This is GetIsMultipleRecordView, a member of class IXMLDisplay.
	GetVisibleFields (see page 2207)	This is GetVisibleFields, a member of class IDataSetFields.
	GetXMLBroker (see page 2201)	Returns the value of the XMLBroker (see page 2203) property.
	GetXMLDataSet (see page 2202)	Returns the value of the XMLDataSet (see page 2203) property.
	GetXMLDisplayName (see page 2230)	This is GetXMLDisplayName, a member of class IXMLDisplay.
	GetXMLRowSetName (see page 2230)	This is GetXMLRowSetName, a member of class IXMLDisplay.
	HasStatusField (see page 2207)	This is HasStatusField, a member of class IDataSetFields.
	IsFieldInUse (see page 2207)	This is IsFieldInUse, a member of class IDataSetFields.
	SetEnableValidateFields (see page 2225)	This is SetEnableValidateFields, a member of class IValidateFields.
	ValidateDataSet (see page 2207)	This is ValidateDataSet, a member of class IDataSetFields.
	ValidateFields (see page 2225)	This is ValidateFields, a member of class IValidateFields.

Public Properties

	Name	Description
R	DataSet (see page 2202)	Provides access to an interface for working with the associated dataset.
	DisplayRows (see page 1817)	Limits the number of rows in the HTML table.
	EnableValidateFields (see page 2226)	This is EnableValidateFields, a member of class IValidateFields.
	HeadingAttributes (see page 1817)	Determines the display characteristics of the table's header.
R	IsMultipleRecordView (see page 2227)	
R	IsMultipleRecordView (see page 2230)	
	RowAttributes (see page 1818)	Determines the display characteristics of the table's body.
	TableAttributes (see page 1818)	Determines the basic display characteristics of the table.
R	VisibleFields (see page 2208)	This is VisibleFields, a member of class IDataSetFields.
R	VisibleFields (see page 2186)	Indicates the controls displayed by the HTML that the XML display group generates.
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.
R	XMLBroker (see page 2203)	Indicates the XML broker that fetches the data packets to which the Web item is bound.
R	XMLDataSet (see page 2203)	Indicates the TXMLDataSet (see page 2164) object that represents the dataset to which the Web item is bound.
R	XMLDisplayName (see page 2231)	This is XMLDisplayName, a member of class IXMLDisplay.
R	XMLRowSetName (see page 2231)	This is XMLRowSetName, a member of class IXMLDisplay.

Published Properties

	Name	Description
	XMLBroker (see page 2187)	Binds the XML display group to an XML broker that represents a dataset.
	XMLDataSetField (see page 2188)	Binds the XML display group to a nested dataset from the XML data packet.

37.1.4.2 TCustomDataGrid.Create Constructor

Creates and initializes an instance of TCustomDataGrid (see page 1811).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomDataGrid(TComponent * AOwner);
```

Description

Call Create to instantiate TCustomDataGrid (see page 1811) at runtime. TCustomDataGrid (see page 1811) components added at design time using the Web Page editor are instantiated automatically.

37.1.4.3 TCustomDataGrid.Destroy Destructor

Frees an instance of TCustomDataGrid (see page 1811).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomDataGrid();
```

Description

Do not call Destroy directly. Instead, call Free, which checks whether the object reference is nil before calling Destroy.

See Also

System::TObject::Free

37.1.4.4 TCustomDataGrid Methods

The methods of the TCustomDataGrid class are listed here.

Protected Methods

	Name	Description
	FormatTable (see page 1814)	This is FormatTable, a member of class TCustomDataGrid.
V	ImplAddElements (see page 1815)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 1815)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1815)	Provides the underlying implementation for the Content method.
V	ImplIsMultipleRecordView (see page 1816)	This is ImplIsMultipleRecordView, a member of class TCustomDataGrid.
	SetHeadingAttributes (see page 1816)	This is SetHeadingAttributes, a member of class TCustomDataGrid.
	SetRowAttributes (see page 1816)	This is SetRowAttributes, a member of class TCustomDataGrid.
	SetTableAttributes (see page 1817)	This is SetTableAttributes, a member of class TCustomDataGrid.

37.1.4.4.1 TCustomDataGrid.FormatTable Method

Pascal

```
function FormatTable(Layout: TLayoutWebContent; Options: TWebContentOptions): string;
```

C++

```
__fastcall UnicodeString FormatTable(TLayoutWebContent Layout, TWebContentOptions Options);
```

Description

This is FormatTable, a member of class TCustomDataGrid.

37.1.4.4.2 TCustomDataGrid.ImplAddElements Method

Provides the underlying implementation for the AddElements method.

Pascal

```
procedure ImplAddElements(AddIntf: IAddScriptElements); override;
```

C++

```
virtual __fastcall ImplAddElements(IAddScriptElements AddIntf);
```

Description

The AddElements method calls ImplAddElements to add any script declarations by calling the AddIntf interface. These declarations allow the generated display control to call script methods in response to user events by ensuring that the required methods are declared.

As implemented in TWebControlGroup, ImplAddElements does nothing. TWebControlGroup descendants override ImplAddElements to add any required declarations.

See Also

AddElements (see page 2066)

37.1.4.4.3 TCustomDataGrid.ImplCanAddClass Method

Provides the underlying implementation for the CanAddClass method.

Pascal

```
function ImplCanAddClass(AParent: TComponent; AClass: TClass): Boolean; override;
```

C++

```
virtual __fastcall Boolean ImplCanAddClass(TComponent * AParent, TClass AClass);
```

Description

The CanAddClass method calls ImpleCanAddClass to determine whether the TWebControlGroup descendant can add a component of type AClass to its WebComponents property list. AParent is the TWebControlGroup instance to which the component will be added.

In TWebControlGroup, ImplCanAddClass always returns true. TWebControlGroup descendants override ImplCanAddClass to indicate what types of Web Items they can use when generating their HTML content.

See Also

CanAddClass (see page 2067)

WebFieldControls (see page 2072)

37.1.4.4.4 TCustomDataGrid.ImplContent Method

Provides the underlying implementation for the Content method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content method calls ImplContent to generate the group of HTML controls. As implemented in TWebControlGroup, ImplContent returns an empty string. TWebControlGroup descendants override it to generate HTML for the group of controls.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the TWebControlGroup should format the HTML controls it generates. It implements the ILayoutWebContent interface.

ImplContent returns the generated HTML.

See Also

Content (see page 2068)

AddElements (see page 2066)

ILayoutWebContent

37.1.4.4.5 TCustomDataGrid.ImplIsMultipleRecordView Method

Pascal

```
function ImplIsMultipleRecordView: Boolean; override;
```

C++

```
virtual __fastcall Boolean ImplIsMultipleRecordView();
```

Description

This is ImplIsMultipleRecordView, a member of class TCustomDataGrid.

37.1.4.4.6 TCustomDataGrid.SetHeadingAttributes Method

Pascal

```
procedure SetHeadingAttributes(const Value: TGridRowAttributes);
```

C++

```
__fastcall SetHeadingAttributes(const TGridRowAttributes Value);
```

Description

This is SetHeadingAttributes, a member of class TCustomDataGrid.

37.1.4.4.7 TCustomDataGrid.SetRowAttributes Method

Pascal

```
procedure SetRowAttributes(const Value: TGridRowAttributes);
```

C++

```
__fastcall SetRowAttributes(const TGridRowAttributes Value);
```

Description

This is SetRowAttributes, a member of class TCustomDataGrid.

37.1.4.4.8 TCustomDataGrid.SetTableAttributes Method

Pascal

```
procedure SetTableAttributes(const Value: TGridAttributes);
```

C++

```
__fastcall SetTableAttributes(const TGridAttributes Value);
```

Description

This is SetTableAttributes, a member of class TCustomDataGrid.

37.1.4.5 TCustomDataGrid Properties

The properties of the TCustomDataGrid class are listed here.

Public Properties

	Name	Description
	DisplayRows (see page 1817)	Limits the number of rows in the HTML table.
	HeadingAttributes (see page 1817)	Determines the display characteristics of the table's header.
	RowAttributes (see page 1818)	Determines the display characteristics of the table's body.
	TableAttributes (see page 1818)	Determines the basic display characteristics of the table.

37.1.4.5.1 TCustomDataGrid.DisplayRows Property

Limits the number of rows in the HTML table.

Pascal

```
property DisplayRows: Integer;
```

C++

```
__property int DisplayRows;
```

Description

Use DisplayRows to specify the number of rows that appear in the generated HTML table.

Note that the number of rows in the HTML table does not limit the number of records that end users can view and edit. When DisplayRows is less than the number of records in the data packet, the control generated by TNextPageButton (see page 1945) or TPriorPageButton (see page 1951) lets end users scroll through the records in the data packet.

See Also

MaxRecords

TNextPageButton (see page 1945)

TPriorPageButton (see page 1951)

37.1.4.5.2 TCustomDataGrid.HeadingAttributes Property

Determines the display characteristics of the table's header.

Pascal

```
property HeadingAttributes: TGridRowAttributes;
```

C++

```
__property TGridRowAttributes HeadingAttributes;
```

Description

Use HeadingAttributes to specify any display attributes that apply only to the heading of the generated HTML table. HeadingAttributes determines the options that are added to the <TR> tag that defines the header of the grid. These include the color of the header, the alignment of the caption for each column, and any custom options or style rules.

Note: To influence the display attributes of the grid as a whole or only the rows of the grid, use the TableAttributes (see page 1818) or RowAttributes (see page 1818) property instead.

See Also

TableAttributes (see page 1818)

RowAttributes (see page 1818)

37.1.4.5.3 TCustomDataGrid.RowAttributes Property

Determines the display characteristics of the table's body.

Pascal

```
property RowAttributes: TGridRowAttributes;
```

C++

```
__property TGridRowAttributes RowAttributes;
```

Description

Use RowAttributes to specify any display attributes that apply only to the rows of the generated HTML table. RowAttributes determines the options that are added to the <TR> tag for each row in the body of the grid. These include the color of the rows, the alignment of data within each cell, and any custom options or style rules.

Note: To influence the display attributes of the grid as a whole or only the header of the grid, use the TableAttributes (see page 1818) or HeadingAttributes (see page 1817) property instead.

See Also

TableAttributes (see page 1818)

HeadingAttributes (see page 1817)

37.1.4.5.4 TCustomDataGrid.TableAttributes Property

Determines the basic display characteristics of the table.

Pascal

```
property TableAttributes: TGridAttributes;
```

C++

```
__property TGridAttributes TableAttributes;
```

Description

Use TableAttributes to specify values for the most common options on the HTML <TABLE> tag. TableAttributes lets you specify the alignment of the table in its HTML document, its color and border size, the offset of data within each table cell, as well as any

custom options or style rules.

Note: To influence the display attributes of only the header of the grid, or only the rows in its body, use the TableAttributes or HeadingAttributes (see page 1817) property instead.

See Also

RowAttributes (see page 1818)

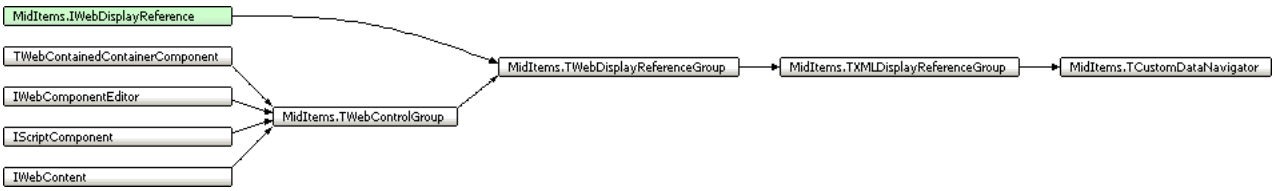
DisplayRows (see page 1817)

HeadingAttributes (see page 1817)

37.1.5 TCustomDataNavigator Class

TCustomDataNavigator assembles the HTML generated by a set of TDataSetButton (see page 1859) and TXMLButton (see page 2151) descendants.

Class Hierarchy



Pascal

```
TCustomDataNavigator = class(TXMLDisplayReferenceGroup);
```

C++

```
class TCustomDataNavigator : public TXMLDisplayReferenceGroup;
```

File

MidItems

Description

Use TCustomDataNavigator as a base class for Web items that generate the HTML to arrange a collection of button controls in a data form. Each TCustomDataNavigator object has an associated Web item that generates display controls in the form. The buttons arranged by the data navigator component manipulate the information these display controls show.

TCustomDataNavigator introduces properties that customize the appearance of the buttons it arranges.

Do not create instances of TCustomDataNavigator. TCustomDataNavigator is intended to act as a base class. Instead, instantiate a descendant class such as TDataNavigator (see page 1855).

See Also

TDataForm (see page 1847)

TFieldGroup (see page 1869)

TDataGrid (see page 1850)

IWebContent

37.1.5.1 TCustomDataNavigator Members

The following tables list the members exposed by TCustomDataNavigator.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	GetDefaultButtons (see page 1821)	This is GetDefaultButtons, a member of class TCustomDataNavigator.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
	GetWebDisplayComponent (see page 2100)	This is GetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.
V	ImplAddElements (see page 2069)	Provides the underlying implementation for the AddElements (see page 2066) method.
V	ImplAddElements (see page 2198)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 1821)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1821)	Provides the underlying implementation for the Content method.
V	ImplGetSubComponents (see page 1822)	Provides the underlying implementation for the GetSubComponents method.
V	Notification (see page 2101)	Forwards notification messages to all owned components.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.
V	SetWebDisplayComponent (see page 2101)	This is SetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.
	VisibleButtons (see page 1822)	This is VisibleButtons, a member of class TCustomDataNavigator.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.
R	WebDisplay (see page 2102)	This is WebDisplay, a member of class TWebDisplayReferenceGroup.
	WebDisplayComponent (see page 2102)	

Public Methods

	Name	Description
V	Create (see page 2065)	Creates and initializes an instance of TWebControlGroup (see page 2064).
V	Create (see page 2100)	Creates an instance of TWebDisplayReferenceGroup (see page 2098).
V	Destroy (see page 2066)	Frees an instance of TWebControlGroup (see page 2064).
V	Destroy (see page 2100)	Frees an instance of TWebDisplayReferenceGroup (see page 2098).
	GetWebDisplayComponent (see page 2228)	This is GetWebDisplayComponent, a member of class IWebDisplayReference.

Public Properties

	Name	Description
	Custom (see page 1823)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1823)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1824)	Applies a style in the page producer's style sheet to the group of buttons.
R	WebDisplayComponent (see page 2228)	
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.
	XMLComponent (see page 2199)	Associates the group with a component that generates HTML display controls.

37.1.5.2 TCustomDataNavigator Methods

The methods of the TCustomDataNavigator class are listed here.

Protected Methods

	Name	Description
	GetDefaultButtons (see page 1821)	This is GetDefaultButtons, a member of class TCustomDataNavigator.
	ImplCanAddClass (see page 1821)	Provides the underlying implementation for the CanAddClass method.
	ImplContent (see page 1821)	Provides the underlying implementation for the Content method.
	ImplGetSubComponents (see page 1822)	Provides the underlying implementation for the GetSubComponents method.
	VisibleButtons (see page 1822)	This is VisibleButtons, a member of class TCustomDataNavigator.

37.1.5.2.1 TCustomDataNavigator.GetDefaultButtons Method

Pascal

```
procedure GetDefaultButtons; virtual;
```

C++

```
virtual __fastcall GetDefaultButtons();
```

Description

This is GetDefaultButtons, a member of class TCustomDataNavigator.

37.1.5.2.2 TCustomDataNavigator.ImplCanAddClass Method

Provides the underlying implementation for the CanAddClass method.

Pascal

```
function ImplCanAddClass(AParent: TComponent; AClass: TClass): Boolean; override;
```

C++

```
virtual __fastcall Boolean ImplCanAddClass(TComponent * AParent, TClass AClass);
```

Description

The CanAddClass method calls ImpleCanAddClass to determine whether the TWebControlGroup descendant can add a component of type AClass to its WebComponents property list. AParent is the TWebControlGroup instance to which the component will be added.

In TWebControlGroup, ImplCanAddClass always returns true. TWebControlGroup descendants override ImplCanAddClass to indicate what types of Web Items they can use when generating their HTML content.

See Also

CanAddClass (see page 2067)

WebFieldControls (see page 2072)

37.1.5.2.3 TCustomDataNavigator.ImplContent Method

Provides the underlying implementation for the Content method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content method calls ImplContent to generate the group of HTML controls. As implemented in TWebControlGroup, ImplContent returns an empty string. TWebControlGroup descendants override it to generate HTML for the group of controls.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the TWebControlGroup should format the HTML controls it generates. It implements the ILayoutWebContent interface.

ImplContent returns the generated HTML.

See Also

Content (see page 2068)

AddElements (see page 2066)

ILayoutWebContent

37.1.5.2.4 TCustomDataNavigator.ImplGetSubComponents Method

Provides the underlying implementation for the GetSubComponents method.

Pascal

```
function ImplGetSubComponents: TObject; override;
```

C++

```
virtual __fastcall TObject * ImplGetSubComponents();
```

Description

The GetSubComponents method calls ImplGetSubComponents to retrieve the value of WebFieldControls as a TObject. TWebControlGroup descendants override this method if the list is not simply the value of WebFieldControls. For example, some TWebControlGroup descendants supply a default value (DefaultWebComponents) when the WebFieldControls property list is empty.

ImplGetSubComponents returns the TWebComponentList object that contains all subitems of the Web control group.

See Also

GetSubComponents (see page 2069)

DefaultWebComponents (see page 2072)

WebFieldControls (see page 2072)

37.1.5.2.5 TCustomDataNavigator.VisibleButtons Method

Pascal

```
function VisibleButtons: TWebComponentList;
```

C++

```
__fastcall TWebComponentList VisibleButtons();
```

Description

This is VisibleButtons, a member of class TCustomDataNavigator.

37.1.5.3 TCustomDataNavigator Properties

The properties of the TCustomDataNavigator class are listed here.

Public Properties

	Name	Description
	Custom (see page 1823)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1823)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1824)	Applies a style in the page producer's style sheet to the group of buttons.

37.1.5.3.1 TCustomDataNavigator.Custom Property

Represents a tag option and its value in the generated TABLE tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TCustomDataNavigator (see page 1819) generates an HTML table to lay out the buttons listed in the WebFieldControls (see page 2072) property. Use Custom to add options that qualify the TABLE tag. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to add a border around the buttons and color the background blue, you can set Custom to

```
BORDER=2 BGCOLOR=BLUE
```

See Also

Style (see page 1823)

StyleRule (see page 1824)

WebFieldControls (see page 2072)

37.1.5.3.2 TCustomDataNavigator.Style Property

Adds style attributes to the generated HTML table.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

TCustomDataNavigator (see page 1819) generates an HTML table to lay out the buttons listed in the WebFieldControls (see page 2072) property. Use Style to add a STYLE option that qualifies the TABLE tag. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated TABLE tag as the value of a STYLE option.

See Also

Custom (see page 1823)

StyleRule (see page 1824)

WebFieldControls (see page 2072)

37.1.5.3.3 TCustomDataNavigator.StyleRule Property

Applies a style in the page producer's style sheet to the group of buttons.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

TCustomDataNavigator (see page 1819) generates an HTML table to lay out the buttons listed in the WebFieldControls (see page 2072) property. Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated table. The value of StyleRule specifies one of the user-defined style names defined in the InternetExpress page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the Style (see page 1823) property.

See Also

Style (see page 1823)

Styles (see page 2272)

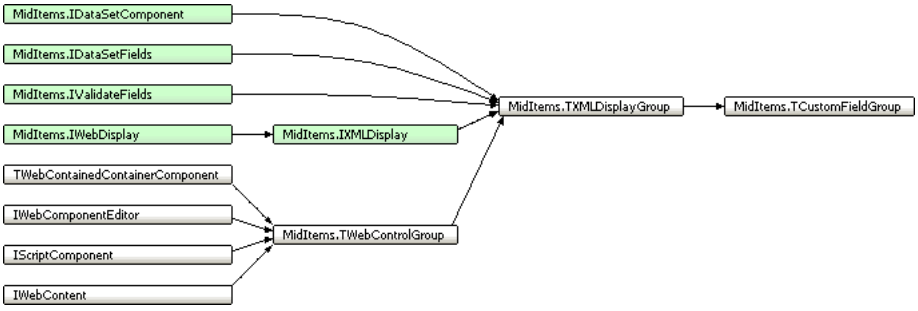
StylesFile (see page 2273)

WebFieldControls (see page 2072)

37.1.6 TCustomFieldGroup Class

TCustomFieldGroup is the base class for Web items that arrange information on an HTML form as a set of labeled fields.

Class Hierarchy



Pascal

```
TCustomFieldGroup = class(TXMLDisplayGroup);
```

C++

```
class TCustomFieldGroup : public TXMLDisplayGroup;
```

File

MidItems

Description

Use TCustomFieldGroup as a base class for Web items that display information in labeled fields on an HTML form. Each TCustomFieldGroup object maintains a list of Web items that generate the individual display fields. These Web items are restricted to classes that are appropriate for the specific TCustomFieldGroup descendant.

If the controls generated for TCustomFieldGroup by its Web items represent database information, you can bind the field group to an XML data packet or dataset field within an XML data packet. The Web items can use that binding to determine the source of the data they display.

TCustomFieldGroup introduces a number of properties that affect the appearance of the generated set of fields.

In order to let other Web items use the HTML that the TCustomFieldGroup object generates, TCustomFieldGroup inherits the IWebContent interface from TWebControlGroup (see page 2064).

Do not create instances of TCustomFieldGroup. Instead, use a descendant of TCustomFieldGroup such as TFieldGroup (see page 1869) or TQueryFieldGroup (see page 1965).

See Also

TFieldGroup (see page 1869)

IWebContent

TQueryFieldGroup (see page 1965)

37.1.6.1 TCustomFieldGroup Members

The following tables list the members exposed by TCustomFieldGroup.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	CreateDefaultFields (see page 2179)	This is CreateDefaultFields, a member of class TXMLDisplayGroup.
	FindField (see page 2180)	This is FindField, a member of class TXMLDisplayGroup.
	GetDataSet (see page 2180)	Returns the value of the DataSet property.
	GetDataSetField (see page 2180)	This is GetDataSetField, a member of class TXMLDisplayGroup.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
	GetEnableValidateFields (see page 2180)	This is GetEnableValidateFields, a member of class TXMLDisplayGroup.
V	GetFieldsList (see page 2181)	This is GetFieldsList, a member of class TXMLDisplayGroup.
	GetIsMultipleRecordView (see page 2181)	This is GetIsMultipleRecordView, a member of class TXMLDisplayGroup.
V	GetLayoutAttributes (see page 1828)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
	GetVisibleFields (see page 2181)	This is GetVisibleFields, a member of class TXMLDisplayGroup.
	GetXMLBroker (see page 2181)	Returns the value of the XMLBroker property.
	GetXMLDataSet (see page 2182)	Returns the value of the XMLDataSet property.

	GetXMLDisplayName (see page 2182)	This is GetXMLDisplayName, a member of class TXMLDisplayGroup.
	GetXMLRowSetName (see page 2182)	This is GetXMLRowSetName, a member of class TXMLDisplayGroup.
	HasStatusField (see page 2182)	This is HasStatusField, a member of class TXMLDisplayGroup.
V	ImplAddElements (see page 2069)	Provides the underlying implementation for the AddElements (see page 2066) method.
V	ImplAddElements (see page 2183)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 2070)	Provides the underlying implementation for the CanAddClass (see page 2067) method.
V	ImplContent (see page 1828)	Provides the underlying implementation for the Content method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	ImplGetSubComponents (see page 2183)	Provides the underlying implementation for the GetSubComponents method.
V	ImplIsMultipleRecordView (see page 2183)	This is ImplIsMultipleRecordView, a member of class TXMLDisplayGroup.
	IsFieldInUse (see page 2184)	This is IsFieldInUse, a member of class TXMLDisplayGroup.
V	Notification (see page 2184)	Forwards notification messages to all owned components.
	SetDataSetField (see page 2184)	This is SetDataSetField, a member of class TXMLDisplayGroup.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.
	SetEnableValidateFields (see page 2185)	This is SetEnableValidateFields, a member of class TXMLDisplayGroup.
	SetXMLBroker (see page 2185)	This is SetXMLBroker, a member of class TXMLDisplayGroup.
	ValidateDataSet (see page 2185)	This is ValidateDataSet, a member of class TXMLDisplayGroup.
	ValidateFields (see page 2185)	This is ValidateFields, a member of class TXMLDisplayGroup.
V	XMLDataChange (see page 2185)	This is XMLDataChange, a member of class TXMLDisplayGroup.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.
	EnableValidateFields (see page 2186)	This is EnableValidateFields, a member of class TXMLDisplayGroup.
R	XMLDataSet (see page 2187)	Indicates the TXMLDataSet object that represents the dataset to which the Web item is bound.

Public Methods

	Name	Description
V	Create (see page 1827)	Creates and initializes an instance of TCustomFieldGroup (see page 1824).
V	Destroy (see page 1827)	Frees an instance of TCustomFieldGroup (see page 1824).
	GetDataSet (see page 2201)	Returns the value of the DataSet (see page 2202) property.
	GetEnableValidateFields (see page 2225)	This is GetEnableValidateFields, a member of class IValidateFields.
	GetFieldsList (see page 2207)	This is GetFieldsList, a member of class IDataSetFields.
	GetIsMultipleRecordView (see page 2227)	This is GetIsMultipleRecordView, a member of class IWebDisplay.
	GetIsMultipleRecordView (see page 2229)	This is GetIsMultipleRecordView, a member of class IXMLDisplay.
	GetVisibleFields (see page 2207)	This is GetVisibleFields, a member of class IDataSetFields.
	GetXMLBroker (see page 2201)	Returns the value of the XMLBroker (see page 2203) property.
	GetXMLDataSet (see page 2202)	Returns the value of the XMLDataSet (see page 2203) property.
	GetXMLDisplayName (see page 2230)	This is GetXMLDisplayName, a member of class IXMLDisplay.
	GetXMLRowSetName (see page 2230)	This is GetXMLRowSetName, a member of class IXMLDisplay.
	HasStatusField (see page 2207)	This is HasStatusField, a member of class IDataSetFields.
	IsFieldInUse (see page 2207)	This is IsFieldInUse, a member of class IDataSetFields.
	SetEnableValidateFields (see page 2225)	This is SetEnableValidateFields, a member of class IValidateFields.
	ValidateDataSet (see page 2207)	This is ValidateDataSet, a member of class IDataSetFields.
	ValidateFields (see page 2225)	This is ValidateFields, a member of class IValidateFields.

Public Properties

	Name	Description
	Custom (see page 1829)	Represents a tag option and its value in the generated TABLE tag.
R	DataSet (see page 2202)	Provides access to an interface for working with the associated dataset.
	EnableValidateFields (see page 2226)	This is EnableValidateFields, a member of class IValidateFields.

	IsMultipleRecordView (see page 2227)	
	IsMultipleRecordView (see page 2230)	
	Style (see page 1829)	Adds style attributes to the generated set of fields.
	StyleRule (see page 1830)	Applies a style in the page producer's style sheet to the table.
	VisibleFields (see page 2208)	This is VisibleFields, a member of class IDataSetFields.
	VisibleFields (see page 2186)	Indicates the controls displayed by the HTML that the XML display group generates.
	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.
	XMLBroker (see page 2203)	Indicates the XML broker that fetches the data packets to which the Web item is bound.
	XMLDataSet (see page 2203)	Indicates the TXMLDataSet (see page 2164) object that represents the dataset to which the Web item is bound.
	XMLDisplayName (see page 2231)	This is XMLDisplayName, a member of class IXMLDisplay.
	XMLRowSetName (see page 2231)	This is XMLRowSetName, a member of class IXMLDisplay.

Published Properties

	Name	Description
	XMLBroker (see page 2187)	Binds the XML display group to an XML broker that represents a dataset.
	XMLDataSetField (see page 2188)	Binds the XML display group to a nested dataset from the XML data packet.

37.1.6.2 TCustomFieldGroup.Create Constructor

Creates and initializes an instance of TCustomFieldGroup (see page 1824).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomFieldGroup(TComponent * AOwner);
```

Description

Call Create to instantiate TCustomFieldGroup (see page 1824) at runtime. TCustomFieldGroup (see page 1824) components added at design time using the Web Page editor are instantiated automatically.

37.1.6.3 TCustomFieldGroup.Destroy Destructor

Frees an instance of TCustomFieldGroup (see page 1824).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomFieldGroup();
```

Description

Do not call Destroy directly. Instead, call Free, which checks whether the object reference is not nil before calling Destroy.

See Also

System::TObject::Free

37.1.6.4 TCustomFieldGroup Methods

The methods of the TCustomFieldGroup class are listed here.

Protected Methods

	Name	Description
	GetLayoutAttributes (see page 1828)	Returns an object that specifies how the generated group of controls should look.
	ImplContent (see page 1828)	Provides the underlying implementation for the Content method.

37.1.6.4.1 TCustomFieldGroup.GetLayoutAttributes Method

Returns an object that specifies how the generated group of controls should look.

Pascal

```
function GetLayoutAttributes: TLayoutAttributes; override;
```

C++

```
virtual __fastcall TLayoutAttributes GetLayoutAttributes();
```

Description

GetLayoutAttributes is used internally by Web items to allow the layout object supplied by a parent (as a parameter to the Content method) to take into account layout properties of the child component.

As implemented by TWebControlGroup, GetLayoutAttributes returns nil (Delphi) or NULL (C++), because TWebControlGroup descendants not make use of layout attributes.

See Also

Content (see page 2068)

37.1.6.4.2 TCustomFieldGroup.ImplContent Method

Provides the underlying implementation for the Content method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content method calls ImplContent to generate the group of HTML controls. As implemented in TWebControlGroup, ImplContent returns an empty string. TWebControlGroup descendants override it to generate HTML for the group of controls.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the TWebControlGroup should format the HTML controls it generates. It implements the ILayoutWebContent interface.

ImplContent returns the generated HTML.

See Also

Content (see page 2068)

AddElements (see page 2066)

ILayoutWebContent

37.1.6.5 TCustomFieldGroup Properties

The properties of the TCustomFieldGroup class are listed here.

Public Properties

	Name	Description
	Custom (see page 1829)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1829)	Adds style attributes to the generated set of fields.
	StyleRule (see page 1830)	Applies a style in the page producer's style sheet to the table.

37.1.6.5.1 TCustomFieldGroup.Custom Property

Represents a tag option and its value in the generated TABLE tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TCustomFieldGroup (see page 1824) generates an HTML table to manage the layout of the HTML controls that its Web items generate. Use Custom to add options that qualify the TABLE tag. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to surround the set of fields with a border and give them a blue background, set Custom to

```
BORDER=2 BGCOLOR=BLUE
```

See Also

Style (see page 1829)

StyleRule (see page 1830)

37.1.6.5.2 TCustomFieldGroup.Style Property

Adds style attributes to the generated set of fields.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

TCustomFieldGroup (see page 1824) generates an HTML table to lay out the fields that its Web items generate. Use Style to add a STYLE option that qualifies the TABLE tag. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated TABLE tag as the value of a STYLE option.

See Also

Custom (see page 1829)

StyleRule (see page 1830)

37.1.6.5.3 TCustomFieldGroup.StyleRule Property

Applies a style in the page producer's style sheet to the table.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated set of fields. The value of StyleRule specifies one of the user-defined style names defined in the InternetExpress page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the Style (see page 1829) property.

See Also

Style (see page 1829)

Styles (see page 2272)

StylesFile (see page 2273)

37.1.7 TCustomLayoutGroup Class

TCustomLayoutGroup is the base class for Web items that arrange content produced by arbitrary Web items.

Class Hierarchy

Pascal

```
TCustomLayoutGroup = class(TWebControlGroup);
```

C++

```
class TCustomLayoutGroup : public TWebControlGroup;
```

File

MidItems

Description

Use TCustomLayoutGroup as a base class for Web items that are used purely for layout. There are few restrictions on what Web items can use a layout group because it has no special content that it displays. It merely imposes a layout on content generated

by other Web items.

TCustomLayoutGroup maintains a list of Web items that display information in the final HTML document, but imposes no restrictions of its own on the possible types of Web items. Instead, it restricts those Web items to the classes that its parent allows. In order to let the parent component use the HTML that TCustomLayoutGroup generates, TCustomLayoutGroup inherits the IWebContent interface from TWebControlGroup (see page 2064).

Do not create instances of TCustomLayoutGroup. Instead, use a descendant of TCustomLayoutGroup such as TLayoutGroup (see page 1939).

See Also

TLayoutGroup (see page 1939)

IWebContent

37.1.7.1 TCustomLayoutGroup Members

The following tables list the members exposed by TCustomLayoutGroup.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
V	GetLayoutAttributes (see page 1832)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
V	ImplAddElements (see page 2069)	Provides the underlying implementation for the AddElements (see page 2066) method.
V	ImplCanAddClass (see page 1833)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1833)	Provides the underlying implementation for the Content method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.

Public Methods

	Name	Description
V	Create (see page 1832)	Creates and initializes an instance of TCustomLayoutGroup (see page 1830).
V	Destroy (see page 1832)	Frees an instance of TCustomLayoutGroup (see page 1830).

Public Properties

	Name	Description
	Custom (see page 1834)	Represents a tag option and its value in the generated TABLE tag.
	DisplayColumns (see page 1834)	Specifies the number of columns the layout group uses to lay out its child controls.
	Style (see page 1835)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1835)	Applies a style in the page producer's style sheet to the group of buttons.
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.

37.1.7.2 TCustomLayoutGroup.Create Constructor

Creates and initializes an instance of TCustomLayoutGroup (see page 1830).

Pascal

```
constructor Create(AComponent: TComponent); override;
```

C++

```
virtual __fastcall TCustomLayoutGroup(TComponent * AComponent);
```

Description

Call Create to instantiate TCustomLayoutGroup (see page 1830) at runtime. TCustomLayoutGroup (see page 1830) components added at design time using the Web Page editor are instantiated automatically.

37.1.7.3 TCustomLayoutGroup.Destroy Destructor

Frees an instance of TCustomLayoutGroup (see page 1830).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomLayoutGroup();
```

Description

Do not call Destroy directly. Instead, call Free, which checks whether the layout group is nil before calling Destroy.

See Also

System::TObject::Free

37.1.7.4 TCustomLayoutGroup Methods

The methods of the TCustomLayoutGroup class are listed here.

Protected Methods

	Name	Description
	GetLayoutAttributes (see page 1832)	Returns an object that specifies how the generated group of controls should look.
	ImplCanAddClass (see page 1833)	Provides the underlying implementation for the CanAddClass method.
	ImplContent (see page 1833)	Provides the underlying implementation for the Content method.

37.1.7.4.1 TCustomLayoutGroup.GetLayoutAttributes Method

Returns an object that specifies how the generated group of controls should look.

Pascal

```
function GetLayoutAttributes: TLayoutAttributes; override;
```

C++

```
virtual __fastcall TLayoutAttributes GetLayoutAttributes();
```

Description

GetLayoutAttributes is used internally by Web items to allow the layout object supplied by a parent (as a parameter to the Content method) to take into account layout properties of the child component.

As implemented by TWebControlGroup, GetLayoutAttributes returns nil (Delphi) or NULL (C++), because TWebControlGroup descendants not make use of layout attributes.

See Also

Content (see page 2068)

37.1.7.4.2 TCustomLayoutGroup.ImplCanAddClass Method

Provides the underlying implementation for the CanAddClass method.

Pascal

```
function ImplCanAddClass(AParent: TComponent; AClass: TClass): Boolean; override;
```

C++

```
virtual __fastcall Boolean ImplCanAddClass(TComponent * AParent, TClass AClass);
```

Description

The CanAddClass method calls ImpleCanAddClass to determine whether the TWebControlGroup descendant can add a component of type AClass to its WebComponents property list. AParent is the TWebControlGroup instance to which the component will be added.

In TWebControlGroup, ImplCanAddClass always returns true. TWebControlGroup descendants override ImplCanAddClass to indicate what types of Web Items they can use when generating their HTML content.

See Also

CanAddClass (see page 2067)

WebFieldControls (see page 2072)

37.1.7.4.3 TCustomLayoutGroup.ImplContent Method

Provides the underlying implementation for the Content method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content method calls ImplContent to generate the group of HTML controls. As implemented in TWebControlGroup, ImplContent returns an empty string. TWebControlGroup descendants override it to generate HTML for the group of controls.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the TWebControlGroup should format the HTML controls it generates. It implements the ILayoutWebContent interface.

ImplContent returns the generated HTML.

See Also

Content (see page 2068)

AddElements (see page 2066)

ILayoutWebContent

37.1.7.5 TCustomLayoutGroup Properties

The properties of the TCustomLayoutGroup class are listed here.

Public Properties

	Name	Description
	Custom (see page 1834)	Represents a tag option and its value in the generated TABLE tag.
	DisplayColumns (see page 1834)	Specifies the number of columns the layout group uses to lay out its child controls.
	Style (see page 1835)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1835)	Applies a style in the page producer's style sheet to the group of buttons.

37.1.7.5.1 TCustomLayoutGroup.Custom Property

Represents a tag option and its value in the generated TABLE tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TCustomLayoutGroup (see page 1830) generates an HTML table to lay out the controls generated by the Web items in its WebFieldControls (see page 2072) property. Use Custom to add options that qualify the TABLE tag. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to add a border around the group of controls and color the background blue, you can set Custom to

```
BORDER=2 BGCOLOR=BLUE
```

See Also

Style (see page 1835)

StyleRule (see page 1835)

WebFieldControls (see page 2072)

DisplayColumns (see page 1834)

37.1.7.5.2 TCustomLayoutGroup.DisplayColumns Property

Specifies the number of columns the layout group uses to lay out its child controls.

Pascal

```
property DisplayColumns: Integer;
```

C++

```
__property int DisplayColumns;
```

Description

TCustomLayoutGroup (see page 1830) generates an HTML table to lay out the controls generated by the Web items in its WebFieldControls (see page 2072) property. It adds the HTML generated by each Web item listed in WebFieldControls (see page 2072) to a cell of the table, filling the cells out row by row until it displays all the Web items. DisplayColumns determines the number of cells in each row.

See Also

WebFieldControls (see page 2072)

37.1.7.5.3 TCustomLayoutGroup.Style Property

Adds style attributes to the generated HTML table.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

TCustomLayoutGroup (see page 1830) generates an HTML table to lay out the controls generated by the Web items in WebFieldControls (see page 2072). Use Style to add a STYLE option that qualifies the TABLE tag. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated TABLE tag as the value of a STYLE option.

See Also

Custom (see page 1834)

StyleRule (see page 1835)

WebFieldControls (see page 2072)

37.1.7.5.4 TCustomLayoutGroup.StyleRule Property

Applies a style in the page producer's style sheet to the group of buttons.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

TCustomLayoutGroup (see page 1830) generates an HTML table to lay out the controls generated by its Web items. Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated table. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the Style (see page 1835) property.

See Also

Style (see page 1835)

Styles

StylesFile

WebFieldControls (see page 2072)

37.1.8 TCustomQueryButtons Class

TCustomQueryButtons assembles the HTML generated by a set of TQueryButton (see page 1953) descendants.

Class Hierarchy

Pascal

```
TCustomQueryButtons = class(TWebControlGroup);
```

C++

```
class TCustomQueryButtons : public TWebControlGroup;
```

File

MidItems

Description

Use TCustomQueryButtons as a base class for Web items that generate the HTML to arrange a collection of button controls in a query form. The buttons arranged by the query buttons component manipulate information shown by display controls in the parent Query form.

TCustomQueryButtons introduces properties that customize the appearance of the buttons it arranges.

Do not create instances of TCustomQueryButtons. TCustomQueryButtons is intended to act as a base class. Instead, instantiate a descendant class such as TQueryButtons (see page 1962).

See Also

TQueryForm (see page 1971)

TQueryFieldGroup (see page 1965)

IWebContent

37.1.8.1 TCustomQueryButtons Members

The following tables list the members exposed by TCustomQueryButtons.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	GetDefaultButtons (see page 1837)	This is GetDefaultButtons, a member of class TCustomQueryButtons.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
V	ImplAddElements (see page 1838)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 1838)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1838)	Provides the underlying implementation for the Content method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.
	VisibleButtons (see page 1839)	This is VisibleButtons, a member of class TCustomQueryButtons.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.

Public Methods

	Name	Description
V	Create (see page 2065)	Creates and initializes an instance of TWebControlGroup (see page 2064).
V	Destroy (see page 2066)	Frees an instance of TWebControlGroup (see page 2064).

Public Properties

	Name	Description
	Custom (see page 1839)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1840)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1840)	Applies a style in the page producer's style sheet to the group of buttons.
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.

37.1.8.2 TCustomQueryButtons Methods

The methods of the TCustomQueryButtons class are listed here.

Protected Methods

	Name	Description
V	GetDefaultButtons (see page 1837)	This is GetDefaultButtons, a member of class TCustomQueryButtons.
V	ImplAddElements (see page 1838)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 1838)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1838)	Provides the underlying implementation for the Content method.
	VisibleButtons (see page 1839)	This is VisibleButtons, a member of class TCustomQueryButtons.

37.1.8.2.1 TCustomQueryButtons.GetDefaultButtons Method

Pascal

```
procedure GetDefaultButtons; virtual;
```

C++

```
virtual __fastcall GetDefaultButtons();
```

Description

This is GetDefaultButtons, a member of class TCustomQueryButtons.

37.1.8.2.2 TCustomQueryButtons.ImplAddElements Method

Provides the underlying implementation for the AddElements method.

Pascal

```
procedure ImplAddElements(AddIntf: IAddScriptElements); override;
```

C++

```
virtual __fastcall ImplAddElements(IAddScriptElements AddIntf);
```

Description

The AddElements method calls ImplAddElements to add any script declarations by calling the AddIntf interface. These declarations allow the generated display control to call script methods in response to user events by ensuring that the required methods are declared.

As implemented in TWebControlGroup, ImplAddElements does nothing. TWebControlGroup descendants override ImplAddElements to add any required declarations.

See Also

AddElements (see page 2066)

37.1.8.2.3 TCustomQueryButtons.ImplCanAddClass Method

Provides the underlying implementation for the CanAddClass method.

Pascal

```
function ImplCanAddClass(AParent: TComponent; AClass: TClass): Boolean; override;
```

C++

```
virtual __fastcall Boolean ImplCanAddClass(TComponent * AParent, TClass AClass);
```

Description

The CanAddClass method calls ImpleCanAddClass to determine whether the TWebControlGroup descendant can add a component of type AClass to its WebComponents property list. AParent is the TWebControlGroup instance to which the component will be added.

In TWebControlGroup, ImplCanAddClass always returns true. TWebControlGroup descendants override ImplCanAddClass to indicate what types of Web Items they can use when generating their HTML content.

See Also

CanAddClass (see page 2067)

WebFieldControls (see page 2072)

37.1.8.2.4 TCustomQueryButtons.ImplContent Method

Provides the underlying implementation for the Content method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content method calls ImplContent to generate the group of HTML controls. As implemented in TWebControlGroup, ImplContent returns an empty string. TWebControlGroup descendants override it to generate HTML for the group of controls.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the TWebControlGroup should format the HTML controls it generates. It implements the ILayoutWebContent interface.

ImplContent returns the generated HTML.

See Also

Content (see page 2068)

AddElements (see page 2066)

ILayoutWebContent

37.1.8.2.5 TCustomQueryButtons.VisibleButtons Method

Pascal

```
function VisibleButtons: TWebComponentList;
```

C++

```
__fastcall TWebComponentList VisibleButtons();
```

Description

This is VisibleButtons, a member of class TCustomQueryButtons.

37.1.8.3 TCustomQueryButtons Properties

The properties of the TCustomQueryButtons class are listed here.

Public Properties

	Name	Description
	Custom (see page 1839)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1840)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1840)	Applies a style in the page producer's style sheet to the group of buttons.

37.1.8.3.1 TCustomQueryButtons.Custom Property

Represents a tag option and its value in the generated TABLE tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TCustomQueryButtons (see page 1836) generates an HTML table to lay out the buttons listed in the WebFieldControls (see page 2072) property. Use Custom to add options that qualify the TABLE tag. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to add a border around the query buttons and color the background blue, you can set Custom to

```
BORDER=2 BGCOLOR=BLUE
```

See Also

Style (see page 1840)

StyleRule (see page 1840)

WebFieldControls (see page 2072)

37.1.8.3.2 TCustomQueryButtons.Style Property

Adds style attributes to the generated HTML table.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

TCustomQueryButtons (see page 1836) generates an HTML table to lay out the buttons listed in the WebFieldControls (see page 2072) property. Use Style to add a STYLE option that qualifies the TABLE tag. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated TABLE tag as the value of a STYLE option.

See Also

Custom (see page 1839)

StyleRule (see page 1840)

WebFieldControls (see page 2072)

37.1.8.3.3 TCustomQueryButtons.StyleRule Property

Applies a style in the page producer's style sheet to the group of buttons.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

TCustomQueryButtons (see page 1836) generates an HTML table to lay out the buttons listed in the WebFieldControls (see page 2072) property. Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the

generated table. The value of StyleRule specifies one of the user-defined style names defined in the InternetExpress page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the Style (see page 1840) property.

See Also

Style (see page 1840)

Styles (see page 2272)

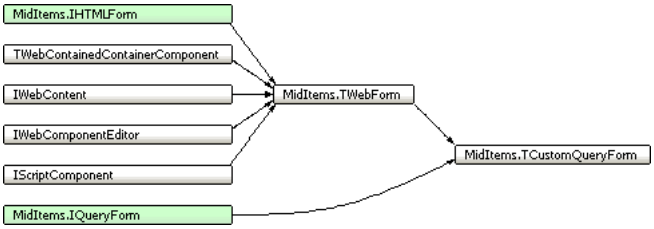
StylesFile (see page 2273)

WebFieldControls (see page 2072)

37.1.9 TCustomQueryForm Class

TCustomQueryForm is the base class for Web items that generate HTML forms for letting users enter application-defined values.

Class Hierarchy



Pascal

```
TCustomQueryForm = class(TWebForm, IQueryForm);
```

C++

```
class TCustomQueryForm : public TWebForm, public IQueryForm;
```

File

MidItems

Description

Use TCustomQueryForm as a base class for components that generate an HTML form that displays application-defined values and submits them to an InternetExpress application. TCustomQueryForm introduces properties for specifying the method and target URL of messages submitted by the form.

When building an HTML form, TCustomQueryForm makes use of content generated by the components listed in its WebComponents (see page 2112) property. These components can include Web items that display application-defined values (TQueryFieldGroup (see page 1965)) or that generate buttons to operate on those values (TQueryButtons (see page 1962)). They can also include components that manage the layout of other HTML controls (TLayoutGroup (see page 1939)).

In order to let an InternetExpress page producer use the generated HTML query form, TCustomQueryForm inherits support for the IWebContent interface from TWebForm (see page 2102).

Do not create instances of TCustomQueryForm. Instead, create descendants of TCustomQueryForm, such as TQueryForm (see page 1971).

See Also

TQueryForm (see page 1971)

IWebContent

TInetXPageProducer (see page 2285)

37.1.9.1 TCustomQueryForm Members

The following tables list the members exposed by TCustomQueryForm.

Protected Methods

	Name	Description
	AddElements (see page 2105)	Used internally to generate any javascript declarations needed by the form.
	AssignStringsCallback (see page 1844)	This is AssignStringsCallback, a member of class TCustomQueryForm.
	CanAddClass (see page 2105)	Specifies whether a Web item of a specified type can be added to the WebComponents (see page 2112) property list.
	Content (see page 2106)	Returns the HTML that defines the HTML form.
V	GetDefaultWebComponents (see page 2106)	Returns nil (Delphi) or NULL (C++).
	GetHTMLFormName (see page 2107)	Returns the name of the generated HTML form.
V	GetHTMLFormTag (see page 1844)	This is GetHTMLFormTag, a member of class TCustomQueryForm.
	GetHTMLFormVarName (see page 2107)	Returns the name of the javascript variable that refers to the generated form.
V	GetLayoutAttributes (see page 2107)	Returns the object that specifies how the generated form should look.
	GetSubComponents (see page 2108)	Returns the value of the WebComponents (see page 2112) property.
	GetVisibleFields (see page 2108)	Returns the value of the VisibleFields (see page 2112) property.
V	ImplAddElements (see page 1844)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 1845)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1845)	Provides the underlying implementation for the Content method.
	MethodString (see page 1846)	This is MethodString, a member of class TCustomQueryForm.
V	SetDefaultWebComponents (see page 2110)	Prevents an application from trying to assign a default set of Web items on the form.

Protected Properties

	Name	Description
R	VisibleFields (see page 2112)	Lists the Web items whose generated content appears in the HTML form.

Public Methods

	Name	Description
	AssignStrings (see page 1843)	Assigns values to display controls from a list of name/value pairs.
V	Create (see page 1843)	Creates and initializes an instance of TCustomQueryForm (see page 1841).
V	Destroy (see page 2104)	Frees an instance of TWebForm (see page 2102).
	GetHTMLFormName (see page 2212)	Returns the name of the generated HTML form.
	GetHTMLFormVarName (see page 2212)	Returns the name of the javascript variable that refers to the generated form.

Public Properties

	Name	Description
	Action (see page 1846)	Specifies the path info portion of the URL to which the query submits request messages.
	Custom (see page 2110)	Represents a tag option and its value in the generated FORM tag.
R	HTMLFormName (see page 2213)	Indicates the name of the generated HTML form.
R	HTMLFormVarName (see page 2213)	Indicates the name of the javascript variable that represents the HTML form.
	Method (see page 1846)	Specifies the METHOD header of messages that the query form submits.
	Style (see page 2111)	Adds style attributes to the generated form.
	StyleRule (see page 2111)	Applies a style in the page producer's style sheet to the button.
R	WebComponents (see page 2112)	Maintains a list of components that generate HTML content for the TWebForm (see page 2102) descendant.

37.1.9.2 TCustomQueryForm.Create Constructor

Creates and initializes an instance of TCustomQueryForm (see page 1841).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomQueryForm(TComponent * AOwner);
```

Description

Call Create to instantiate a query form at runtime. Query forms added at design time using the Web Page editor are instantiated automatically.

37.1.9.3 TCustomQueryForm Methods

The methods of the TCustomQueryForm class are listed here.

Protected Methods

	Name	Description
	AssignStringsCallback (see page 1844)	This is AssignStringsCallback, a member of class TCustomQueryForm.
V	GetHTMLFormTag (see page 1844)	This is GetHTMLFormTag, a member of class TCustomQueryForm.
V	ImplAddElements (see page 1844)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 1845)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1845)	Provides the underlying implementation for the Content method.
	MethodString (see page 1846)	This is MethodString, a member of class TCustomQueryForm.

Public Methods

	Name	Description
	AssignStrings (see page 1843)	Assigns values to display controls from a list of name/value pairs.

37.1.9.3.1 TCustomQueryForm.AssignStrings Method

Assigns values to display controls from a list of name/value pairs.

Pascal

```
procedure AssignStrings(Value: TStrings);
```

C++

```
__fastcall AssignStrings(TStrings Value);
```

Description

Use AssignStrings to initialize the values displayed by the controls in the query form. For each display control named in the specified list of strings, AssignStrings initializes its Text property with the associated value.

Value is a list of strings with the form Name=Value. The Name portion identifies the Web item for a display control that is listed in the WebComponents (see page 2112) property list or is a child of one of those Web items. Name must match the string returned by the component's GetHTMLControlName method. The Value portion of the string specifies the value that is assigned to the Web item's Text property.

Note: AssignStrings is a convenient way to initialize the values in a query form before the HTML document is sent to a Web browser. When the browser returns values that are entered by the end user, the Web server application must extract them from

the request message's ContentFields property.

See Also

ContentFields

WebComponents (see page 2112)

GetHTMLControlName (see page 2078)

Text (see page 1987)

Text (see page 1999)

Text (see page 2010)

37.1.9.3.2 TCustomQueryForm.AssignStringsCallback Method

Pascal

```
procedure AssignStringsCallback(AComponent: TComponent);
```

C++

```
__fastcall AssignStringsCallback(TComponent * AComponent);
```

Description

This is AssignStringsCallback, a member of class TCustomQueryForm.

37.1.9.3.3 TCustomQueryForm.GetHTMLFormTag Method

Pascal

```
function GetHTMLFormTag(Options: TWebContentOptions): string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetHTMLFormTag(TWebContentOptions Options);
```

Description

This is GetHTMLFormTag, a member of class TCustomQueryForm.

37.1.9.3.4 TCustomQueryForm.ImplAddElements Method

Provides the underlying implementation for the AddElements method.

Pascal

```
procedure ImplAddElements(AddIntf: IAddScriptElements); override;
```

C++

```
virtual __fastcall ImplAddElements(IAddScriptElements AddIntf);
```

Description

The AddElements method calls ImplAddElements to add any javascript declarations by calling the AddIntf interface. These declarations allow the generated display control to call javascript methods in response to user events by ensuring that the required methods are declared.

As implemented in TWebForm, ImplAddElements does nothing. TWebForm descendants can override ImplAddElements to add any required declarations.

See Also

AddElements (see page 2105)

37.1.9.3.5 TCustomQueryForm.ImplCanAddClass Method

Provides the underlying implementation for the CanAddClass method.

Pascal

```
function ImplCanAddClass(AParent: TComponent; AClass: TClass): Boolean; override;
```

C++

```
virtual __fastcall Boolean ImplCanAddClass(TComponent * AParent, TClass AClass);
```

Description

The CanAddClass method calls ImpleCanAddClass to determine whether the TWebForm descendant can add a component of type AClass to its WebComponents property list. AParent is the TWebForm instance to which the component will be added.

In TWebForm, ImplCanAddClass is abstract or, in C++ terminology, pure virtual. TWebForm descendants must override ImplCanAddClass to indicate what types of Web Items they can use when generating their HTML forms.

See Also

CanAddClass (see page 2105)

WebComponents (see page 2112)

37.1.9.3.6 TCustomQueryForm.ImplContent Method

Provides the underlying implementation for the Content method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content method calls ImplContent to generate the HTML form. ImplContent is anabstract or, in C++ terminology, pure virtual method in TWebForm. TWebForm descendants override it to generate the HTML form.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the TWebForm should format the HTML control it generates. Typically, this parameter is nil (Delphi) or NULL (C++) for forms.

ImplContent returns an HTML form definition.

See Also

Content (see page 2106)

AddElements (see page 2105)

37.1.9.3.7 TCustomQueryForm.MethodString Method

Pascal

```
function MethodString: string;
```

C++

```
__fastcall UnicodeString MethodString();
```

Description

This is MethodString, a member of class TCustomQueryForm.

37.1.9.4 TCustomQueryForm Properties

The properties of the TCustomQueryForm class are listed here.

Public Properties

	Name	Description
	Action (see page 1846)	Specifies the path info portion of the URL to which the query submits request messages.
	Method (see page 1846)	Specifies the METHOD header of messages that the query form submits.

37.1.9.4.1 TCustomQueryForm.Action Property

Specifies the path info portion of the URL to which the query submits request messages.

Pascal

```
property Action: string;
```

C++

```
__property UnicodeString Action;
```

Description

Use Action to specify the target URL for messages that the query form submits. Action supplies the PathInfo property of Web request messages from the query form. This value is matched to the PathInfo property of a Web action item when determining what object responds to those messages.

See Also

TCustomQueryForm (see page 1841)

PathInfo

PathInfo

37.1.9.4.2 TCustomQueryForm.Method Property

Specifies the METHOD header of messages that the query form submits.

Pascal

```
property Method: TFormMethod;
```

C++

```
__property TFormMethod Method;
```

Description

Set Method to specify the METHOD header for messages that the query form submits. This value gives rise to the MethodType property of the HTTP request message. By matching this to the MethodType property of Web action items, a Web dispatcher determines what objects handle these messages.

See Also

TCustomQueryForm (see page 1841)

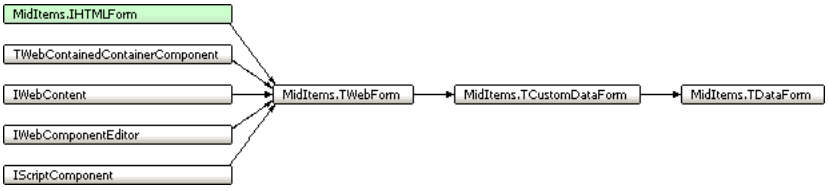
PathInfo

PathInfo

37.1.10 TDataForm Class

TDataForm generates an HTML form that displays database information and lets users apply updates.

Class Hierarchy



Pascal

```
TDataForm = class(TCustomDataForm);
```

C++

```
class TDataForm : public TCustomDataForm;
```

File

MidItems

Description

Add TDataForm to the Web items of an Internet Express page producer (TInetXPageProducer (see page 2285)) to add an HTML form that displays database information that users can view and edit.

Data forms generate an HTML form that contains controls which display information from an XML data packet. Buttons on the form manipulate that data, and submit updates back to the XML broker that supplies the XML data packet.

TDataForm generates the HTML FORM tag and assembles the HTML generated by its Web items. It restricts the Web items that generate the contents of the form to components that generate data displays (TDataGrid (see page 1850) or TFieldGroup (see page 1869)), components that contain buttons which operate on those displays (TDataNavigator (see page 1855)), and components that manage the layout of HTML controls (TLayoutGroup (see page 1939)). At design time, you can add these Web items to the form using the Web page editor.

See Also

TDataGrid (see page 1850)

TBasePageItemsProducer (see page 2966)

TInetXPageProducer (see page 2285)

TFieldGroup (see page 1869)

TLayoutGroup (see page 1939)

TDataNavigator (see page 1855)

TApplyUpdatesButton (see page 1798)

Using the Web Page Editor

37.1.10.1 TDataForm Members

The following tables list the members exposed by TDataForm.

Protected Methods

	Name	Description
	AddElements (see page 2105)	Used internally to generate any javascript declarations needed by the form.
	CanAddClass (see page 2105)	Specifies whether a Web item of a specified type can be added to the WebComponents (see page 2112) property list.
	Content (see page 2106)	Returns the HTML that defines the HTML form.
V	GetDefaultWebComponents (see page 2106)	Returns nil (Delphi) or NULL (C++).
	GetHTMLFormName (see page 2107)	Returns the name of the generated HTML form.
V	GetHTMLFormTag (see page 1810)	This is GetHTMLFormTag, a member of class TCustomDataForm.
	GetHTMLFormVarName (see page 2107)	Returns the name of the javascript variable that refers to the generated form.
V	GetLayoutAttributes (see page 2107)	Returns the object that specifies how the generated form should look.
	GetSubComponents (see page 2108)	Returns the value of the WebComponents (see page 2112) property.
	GetVisibleFields (see page 2108)	Returns the value of the VisibleFields (see page 2112) property.
V	ImplAddElements (see page 2108)	Provides the underlying implementation for the AddElements (see page 2105) method.
V	ImplCanAddClass (see page 1810)	Provides the underlying implementation for the CanAddClass method.
A	ImplCanAddClass (see page 2109)	Provides the underlying implementation for the CanAddClass (see page 2105) method.
V	ImplContent (see page 1810)	Provides the underlying implementation for the Content method.
A	ImplContent (see page 2109)	Provides the underlying implementation for the Content (see page 2106) method.
V	SetDefaultWebComponents (see page 2110)	Prevents an application from trying to assign a default set of Web items on the form.

Protected Properties

	Name	Description
R	VisibleFields (see page 2112)	Lists the Web items whose generated content appears in the HTML form.

Public Methods

	Name	Description
V	Create (see page 2104)	Creates and initializes an instance of TWebForm (see page 2102).
V	Destroy (see page 2104)	Frees an instance of TWebForm (see page 2102).
	GetHTMLFormName (see page 2212)	Returns the name of the generated HTML form.
	GetHTMLFormVarName (see page 2212)	Returns the name of the javascript variable that refers to the generated form.

Public Properties

	Name	Description
R	HTMLFormName (see page 2213)	Indicates the name of the generated HTML form.
R	HTMLFormVarName (see page 2213)	Indicates the name of the javascript variable that represents the HTML form.
R	WebComponents (see page 2112)	Maintains a list of components that generate HTML content for the TWebForm (see page 2102) descendant.

Published Properties

	Name	Description
	Custom (see page 1849)	Represents a tag option and its value in the generated FORM tag.
	Style (see page 1849)	Adds style attributes to the generated form.
	StyleRule (see page 1850)	Applies a style in the page producer's style sheet to the button.

37.1.10.2 TDataForm Properties

The properties of the TDataForm class are listed here.

Published Properties

	Name	Description
	Custom (see page 1849)	Represents a tag option and its value in the generated FORM tag.
	Style (see page 1849)	Adds style attributes to the generated form.
	StyleRule (see page 1850)	Applies a style in the page producer's style sheet to the button.

37.1.10.2.1 TDataForm.Custom Property

Represents a tag option and its value in the generated FORM tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

Use Custom to represent one or more options that qualify the FORM tag generated by the TWebForm descendant. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to specify that the form includes files with the text it submits and indicate that the page returned after submitting the form goes into a separate window, use a Custom value such as the following:

```
ENCTYPE="multipart/form-data" TARGET="ResultFrame"
```

See Also

Style (see page 2111)

37.1.10.2.2 TDataForm.Style Property

Adds style attributes to the generated form.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to add a STYLE option that qualifies the FORM tag generated by the TWebForm descendant. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated FORM tag as the value of a STYLE option.

See Also

Custom (see page 2110)

StyleRule (see page 2111)

37.1.10.2.3 TDataForm.StyleRule Property

Applies a style in the page producer's style sheet to the button.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated form. The value of StyleRule specifies one of the user-defined style names defined in the InternetExpress page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the Style property.

See Also

Style (see page 2111)

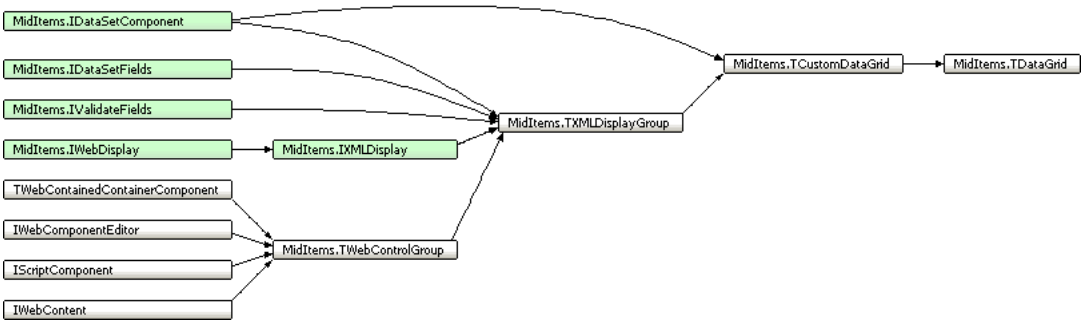
Styles (see page 2272)

StylesFile (see page 2273)

37.1.11 TDataGrid Class

TDataGrid generates an HTML table to display a dataset in an XML data packet.

Class Hierarchy



Pascal

```
TDataGrid = class(TCustomDataGrid);
```

C++

```
class TDataGrid : public TCustomDataGrid;
```

File

MidItems

Description

Add TDataGrid to the Web items of a TDataForm (see page 1847) component (or a layout group that is a child of a TDataForm (see page 1847) component.) TDataGrid generates an HTML table that displays multiple records from a dataset as a set of rows and columns.

Each TDataGrid object maintains a list of Web items that generate the individual fields (columns) of the table. These Web items can be from the following classes: TTextColumn (see page 2049), TSelectOptionColumn, TTextAreaColumn (see page 2041), and TStatusColumn (see page 2033). You can bind the data grid to an XML data packet or dataset field within an XML data packet. The column Web items can then use that binding to determine the source of the data they display.

You can also use the properties of TDataGrid to affect the appearance of the generated table.

See Also

TSelectOptionsColumn (see page 2024)

TStatusColumn (see page 2033)

TTextAreaColumn (see page 2041)

TTextColumn (see page 2049)

TDataForm (see page 1847)

TLayoutGroup (see page 1939)

TWebDataDisplay (see page 2073)

Using the Web Page Editor

37.1.11.1 TDataGrid Members

The following tables list the members exposed by TDataGrid.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	CreateDefaultFields (see page 2179)	This is CreateDefaultFields, a member of class TXMLDisplayGroup.
	FindField (see page 2180)	This is FindField, a member of class TXMLDisplayGroup.
	FormatTable (see page 1814)	This is FormatTable, a member of class TCustomDataGrid.
	GetDataSet (see page 2180)	Returns the value of the DataSet property.
	GetDataSetField (see page 2180)	This is GetDataSetField, a member of class TXMLDisplayGroup.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
	GetEnableValidateFields (see page 2180)	This is GetEnableValidateFields, a member of class TXMLDisplayGroup.
V	GetFieldsList (see page 2181)	This is GetFieldsList, a member of class TXMLDisplayGroup.
	GetIsMultipleRecordView (see page 2181)	This is GetIsMultipleRecordView, a member of class TXMLDisplayGroup.

	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
	GetVisibleFields (see page 2181)	This is GetVisibleFields, a member of class TXMLDisplayGroup.
	GetXMLBroker (see page 2181)	Returns the value of the XMLBroker property.
	GetXMLDataSet (see page 2182)	Returns the value of the XMLDataSet property.
	GetXMLDisplayName (see page 2182)	This is GetXMLDisplayName, a member of class TXMLDisplayGroup.
	GetXMLRowSetName (see page 2182)	This is GetXMLRowSetName, a member of class TXMLDisplayGroup.
	HasStatusField (see page 2182)	This is HasStatusField, a member of class TXMLDisplayGroup.
	ImplAddElements (see page 1815)	Provides the underlying implementation for the AddElements method.
	ImplAddElements (see page 2069)	Provides the underlying implementation for the AddElements (see page 2066) method.
	ImplAddElements (see page 2183)	Provides the underlying implementation for the AddElements method.
	ImplCanAddClass (see page 1815)	Provides the underlying implementation for the CanAddClass method.
	ImplCanAddClass (see page 2070)	Provides the underlying implementation for the CanAddClass (see page 2067) method.
	ImplContent (see page 1815)	Provides the underlying implementation for the Content method.
	ImplContent (see page 2070)	Provides the underlying implementation for the Content (see page 2068) method.
	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
	ImplGetSubComponents (see page 2183)	Provides the underlying implementation for the GetSubComponents method.
	ImplIsMultipleRecordView (see page 1816)	This is ImplIsMultipleRecordView, a member of class TCustomDataGrid.
	ImplIsMultipleRecordView (see page 2183)	This is ImplIsMultipleRecordView, a member of class TXMLDisplayGroup.
	IsFieldInUse (see page 2184)	This is IsFieldInUse, a member of class TXMLDisplayGroup.
	Notification (see page 2184)	Forwards notification messages to all owned components.
	SetDataSetField (see page 2184)	This is SetDataSetField, a member of class TXMLDisplayGroup.
	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.
	SetEnableValidateFields (see page 2185)	This is SetEnableValidateFields, a member of class TXMLDisplayGroup.
	SetHeadingAttributes (see page 1816)	This is SetHeadingAttributes, a member of class TCustomDataGrid.
	SetRowAttributes (see page 1816)	This is SetRowAttributes, a member of class TCustomDataGrid.
	SetTableAttributes (see page 1817)	This is SetTableAttributes, a member of class TCustomDataGrid.
	SetXMLBroker (see page 2185)	This is SetXMLBroker, a member of class TXMLDisplayGroup.
	ValidateDataSet (see page 2185)	This is ValidateDataSet, a member of class TXMLDisplayGroup.
	ValidateFields (see page 2185)	This is ValidateFields, a member of class TXMLDisplayGroup.
	XMLDataChange (see page 2185)	This is XMLDataChange, a member of class TXMLDisplayGroup.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.
	EnableValidateFields (see page 2186)	This is EnableValidateFields, a member of class TXMLDisplayGroup.
	XMLDataSet (see page 2187)	Indicates the TXMLDataSet object that represents the dataset to which the Web item is bound.

Public Methods

	Name	Description
	Create (see page 1814)	Creates and initializes an instance of TCustomDataGrid (see page 1811).
	Create (see page 2065)	Creates and initializes an instance of TWebControlGroup (see page 2064).
	Create (see page 2178)	Creates and initializes an instance of TXMLDisplayGroup (see page 2176).
	Destroy (see page 1814)	Frees an instance of TCustomDataGrid (see page 1811).
	Destroy (see page 2066)	Frees an instance of TWebControlGroup (see page 2064).
	Destroy (see page 2178)	Frees an instance of TXMLDisplayGroup (see page 2176).
	GetDataSet (see page 2201)	Returns the value of the DataSet (see page 2202) property.
	GetEnableValidateFields (see page 2225)	This is GetEnableValidateFields, a member of class IValidateFields.
	GetFieldsList (see page 2207)	This is GetFieldsList, a member of class IDataSetFields.
	GetIsMultipleRecordView (see page 2227)	This is GetIsMultipleRecordView, a member of class IWebDisplay.

	GetIsMultipleRecordView (see page 2229)	This is GetIsMultipleRecordView, a member of class IXMLDisplay.
	GetVisibleFields (see page 2207)	This is GetVisibleFields, a member of class IDataSetFields.
	GetXMLBroker (see page 2201)	Returns the value of the XMLBroker (see page 2203) property.
	GetXMLDataSet (see page 2202)	Returns the value of the XMLDataSet (see page 2203) property.
	GetXMLDisplayName (see page 2230)	This is GetXMLDisplayName, a member of class IXMLDisplay.
	GetXMLRowSetName (see page 2230)	This is GetXMLRowSetName, a member of class IXMLDisplay.
	HasStatusField (see page 2207)	This is HasStatusField, a member of class IDataSetFields.
	IsFieldInUse (see page 2207)	This is IsFieldInUse, a member of class IDataSetFields.
	SetEnableValidateFields (see page 2225)	This is SetEnableValidateFields, a member of class IValidateFields.
	ValidateDataSet (see page 2207)	This is ValidateDataSet, a member of class IDataSetFields.
	ValidateFields (see page 2225)	This is ValidateFields, a member of class IValidateFields.

Public Properties

	Name	Description
R	DataSet (see page 2202)	Provides access to an interface for working with the associated dataset.
	EnableValidateFields (see page 2226)	This is EnableValidateFields, a member of class IValidateFields.
R	IsMultipleRecordView (see page 2227)	
R	IsMultipleRecordView (see page 2230)	
R	VisibleFields (see page 2208)	This is VisibleFields, a member of class IDataSetFields.
R	VisibleFields (see page 2186)	Indicates the controls displayed by the HTML that the XML display group generates.
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.
R	XMLBroker (see page 2203)	Indicates the XML broker that fetches the data packets to which the Web item is bound.
R	XMLDataSet (see page 2203)	Indicates the TXMLDataSet (see page 2164) object that represents the dataset to which the Web item is bound.
R	XMLDisplayName (see page 2231)	This is XMLDisplayName, a member of class IXMLDisplay.
R	XMLRowSetName (see page 2231)	This is XMLRowSetName, a member of class IXMLDisplay.

Published Properties

	Name	Description
	DisplayRows (see page 1853)	Limits the number of rows in the HTML table.
	HeadingAttributes (see page 1854)	Determines the display characteristics of the table's header.
	RowAttributes (see page 1854)	Determines the display characteristics of the table's body.
	TableAttributes (see page 1855)	Determines the basic display characteristics of the table.
	XMLBroker (see page 2187)	Binds the XML display group to an XML broker that represents a dataset.
	XMLDataSetField (see page 2188)	Binds the XML display group to a nested dataset from the XML data packet.

37.1.11.2 TDataGrid Properties

The properties of the TDataGrid class are listed here.

Published Properties

	Name	Description
	DisplayRows (see page 1853)	Limits the number of rows in the HTML table.
	HeadingAttributes (see page 1854)	Determines the display characteristics of the table's header.
	RowAttributes (see page 1854)	Determines the display characteristics of the table's body.
	TableAttributes (see page 1855)	Determines the basic display characteristics of the table.

37.1.11.2.1 TDataGrid.DisplayRows Property

Limits the number of rows in the HTML table.

Pascal

```
property DisplayRows: Integer;
```

C++

```
__property int DisplayRows;
```

Description

Use DisplayRows to specify the number of rows that appear in the generated HTML table.

Note that the number of rows in the HTML table does not limit the number of records that end users can view and edit. When DisplayRows is less than the number of records in the data packet, the control generated by TNextPageButton or TPriorPageButton lets end users scroll through the records in the data packet.

See Also

MaxRecords

TNextPageButton (see page 1945)

TPriorPageButton (see page 1951)

37.1.11.2.2 TDataGrid.HeadingAttributes Property

Determines the display characteristics of the table's header.

Pascal

```
property HeadingAttributes: TGridRowAttributes;
```

C++

```
__property TGridRowAttributes HeadingAttributes;
```

Description

Use HeadingAttributes to specify any display attributes that apply only to the heading of the generated HTML table. HeadingAttributes determines the options that are added to the <TR> tag that defines the header of the grid. These include the color of the header, the alignment of the caption for each column, and any custom options or style rules.

Note: To influence the display attributes of the grid as a whole or only the rows of the grid, use the TableAttributes or RowAttributes property instead.

See Also

TableAttributes (see page 1818)

RowAttributes (see page 1818)

37.1.11.2.3 TDataGrid.RowAttributes Property

Determines the display characteristics of the table's body.

Pascal

```
property RowAttributes: TGridRowAttributes;
```

C++

```
__property TGridRowAttributes RowAttributes;
```

Description

Use RowAttributes to specify any display attributes that apply only to the rows of the generated HTML table. RowAttributes

determines the options that are added to the <TR> tag for each row in the body of the grid. These include the color of the rows, the alignment of data within each cell, and any custom options or style rules.

Note: To influence the display attributes of the grid as a whole or only the header of the grid, use the TableAttributes or HeadingAttributes property instead.

See Also

TableAttributes (see page 1818)

HeadingAttributes (see page 1817)

37.1.11.2.4 TDataGrid.TableAttributes Property

Determines the basic display characteristics of the table.

Pascal

```
property TableAttributes: TGridAttributes;
```

C++

```
__property TGridAttributes TableAttributes;
```

Description

Use TableAttributes to specify values for the most common options on the HTML <TABLE> tag. TableAttributes lets you specify the alignment of the table in its HTML document, its color and border size, the offset of data within each table cell, as well as any custom options or style rules.

Note: To influence the display attributes of only the header of the grid, or only the rows in its body, use the TableAttributes or HeadingAttributes property instead.

See Also

RowAttributes (see page 1818)

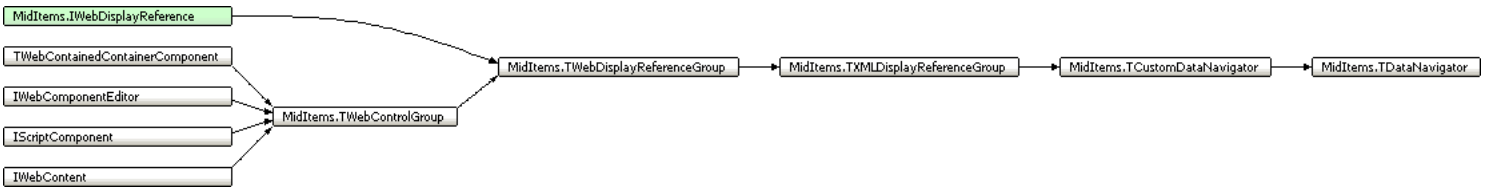
DisplayRows (see page 1817)

HeadingAttributes (see page 1817)

37.1.12 TDataNavigator Class

TDataNavigator assembles the HTML generated by a set of TDataSetButton (see page 1859) and TXMLButton (see page 2151) descendants.

Class Hierarchy



Pascal

```
TDataNavigator = class(TCustomDataNavigator);
```

C++

```
class TDataNavigator : public TCustomDataNavigator;
```

File

MidItems

Description

Add TDataNavigator as the child of a TDataForm (see page 1847) component (or a layout group that is a child of a TDataForm (see page 1847) component) to enable the generated HTML form to navigate the displayed data and apply updates that end users make to the data. TDataNavigator can be associated with the Web item that displays the data, such as TDataGrid (see page 1850) or TFieldGroup (see page 1869).

TDataNavigator generates HTML containing the controls generated by its children. These children must be descendants of TDataSetButton (see page 1859) or TXMLButton (see page 2151), such as TInsertButton (see page 1935) and TApplyUpdatesButton (see page 1798).

Use the properties of TDataNavigator to customize the appearance of the group of buttons.

See Also

TDataForm (see page 1847)

TFieldGroup (see page 1869)

TDataGrid (see page 1850)

Using the Web Page Editor

37.1.12.1 TDataNavigator Members

The following tables list the members exposed by TDataNavigator.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	GetDefaultButtons (see page 1821)	This is GetDefaultButtons, a member of class TCustomDataNavigator.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
	GetWebDisplayComponent (see page 2100)	This is GetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.
V	ImplAddElements (see page 2069)	Provides the underlying implementation for the AddElements (see page 2066) method.
V	ImplAddElements (see page 2198)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 1821)	Provides the underlying implementation for the CanAddClass method.
V	ImplCanAddClass (see page 2070)	Provides the underlying implementation for the CanAddClass (see page 2067) method.
V	ImplContent (see page 1821)	Provides the underlying implementation for the Content method.
V	ImplContent (see page 2070)	Provides the underlying implementation for the Content (see page 2068) method.
V	ImplGetSubComponents (see page 1822)	Provides the underlying implementation for the GetSubComponents method.

	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
	Notification (see page 2101)	Forwards notification messages to all owned components.
	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.
	SetWebDisplayComponent (see page 2101)	This is SetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.
	VisibleButtons (see page 1822)	This is VisibleButtons, a member of class TCustomDataNavigator.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.
R	WebDisplay (see page 2102)	This is WebDisplay, a member of class TWebDisplayReferenceGroup.
	WebDisplayComponent (see page 2102)	

Public Methods

	Name	Description
	Create (see page 2065)	Creates and initializes an instance of TWebControlGroup (see page 2064).
	Create (see page 2100)	Creates an instance of TWebDisplayReferenceGroup (see page 2098).
	Destroy (see page 2066)	Frees an instance of TWebControlGroup (see page 2064).
	Destroy (see page 2100)	Frees an instance of TWebDisplayReferenceGroup (see page 2098).
	GetWebDisplayComponent (see page 2228)	This is GetWebDisplayComponent, a member of class IWebDisplayReference.

Public Properties

	Name	Description
R	WebDisplayComponent (see page 2228)	
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.

Published Properties

	Name	Description
	Custom (see page 1857)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1858)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1858)	Applies a style in the page producer's style sheet to the group of buttons.
	XMLComponent (see page 1859)	Associates the group with a component that generates HTML display controls.

37.1.12.2 TDataNavigator Properties

The properties of the TDataNavigator class are listed here.

Published Properties

	Name	Description
	Custom (see page 1857)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1858)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1858)	Applies a style in the page producer's style sheet to the group of buttons.
	XMLComponent (see page 1859)	Associates the group with a component that generates HTML display controls.

37.1.12.2.1 TDataNavigator.Custom Property

Represents a tag option and its value in the generated TABLE tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TCustomDataNavigator generates an HTML table to lay out the buttons listed in the WebFieldControls property. Use Custom to add options that qualify the TABLE tag. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to add a border around the buttons and color the background blue, you can set Custom to

```
BORDER=2 BGCOLOR=BLUE
```

See Also

Style (see page 1823)

StyleRule (see page 1824)

WebFieldControls (see page 2072)

37.1.12.2.2 TDataNavigator.Style Property

Adds style attributes to the generated HTML table.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

TCustomDataNavigator generates an HTML table to lay out the buttons listed in the WebFieldControls property. Use Style to add a STYLE option that qualifies the TABLE tag. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated TABLE tag as the value of a STYLE option.

See Also

Custom (see page 1823)

StyleRule (see page 1824)

WebFieldControls (see page 2072)

37.1.12.2.3 TDataNavigator.StyleRule Property

Applies a style in the page producer's style sheet to the group of buttons.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

TCustomDataNavigator generates an HTML table to lay out the buttons listed in the WebFieldControls property. Use StyleRule

to assign one of the user-defined styles in the HTML document's style sheet to the generated table. The value of StyleRule specifies one of the user-defined style names defined in the InternetExpress page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the Style property.

See Also

Style (see page 1823)

Styles (see page 2272)

StylesFile (see page 2273)

WebFieldControls (see page 2072)

37.1.12.2.4 TDataNavigator.XMLComponent Property

Associates the group with a component that generates HTML display controls.

Pascal

```
property XMLComponent: TComponent;
```

C++

```
__property TComponent * XMLComponent;
```

Description

Use XMLComponent to associate the group of buttons with a Web item that generates the display controls those buttons influence. Typical values of XMLComponent are TFieldGroup or TDataGrid.

If all the buttons arranged by the TXMLDisplayReferenceGroup object operate on the same set of display controls, it is most efficient to specify the Web item generating these controls as the value of XMLComponent. Then, each button uses this value to determine its associated display controls when its XMLUseParent property is true. Any buttons that do not operate on the display controls generated by XMLComponent can set their own XMLComponent property to another Web item.

See Also

XMLComponent (see page 2195)

XMLUseParent (see page 2196)

TFieldGroup (see page 1869)

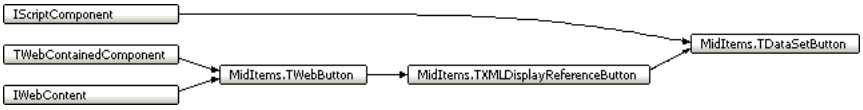
TDataGrid (see page 1850)

TQueryFieldGroup (see page 1965)

37.1.13 TDataSetButton Class

TDataSetButton is the base class for Web items that create HTML buttons for navigating or editing data.

Class Hierarchy



Pascal

```
TDataSetButton = class(TXMLDisplayReferenceButton, IScriptComponent);
```

C++

```
class TDataSetButton : public TXMLDisplayReferenceButton, public IScriptComponent;
```

File

MidItems

Description

Use TDataSetButton as a base class for components that generate the HTML statements defining a specialized button control for working with the data displayed by another Web item. TDataSetButton objects are used by TDataNavigator (see page 1855), which organizes the generated buttons.

TDataSetButton defines properties and methods for

Generating the HTML code to define the button control.

Associating the button with another Web item that generates HTML for displaying the data that the button manipulates.

Specifying tag attributes of the HTML button control, including those defined by a style sheet.

Fitting the component into a hierarchy of Web items, each of which generates part of a larger HTML document.

In order to allow other Web items to use the HTML generated by the button control, TDataSetButton inherits support for the IWebContent interface from TWebButton (see page 2058).

Do not create instances of TDataSetButton. TDataSetButton relies on anabstract or, in C++ terminology, pure virtual method to generate the HTML statements defining a button control. Instead, create descendants of TDataSetButton, such as TNextButton (see page 1943), TPostButton (see page 1947), TInsertButton (see page 1935), and so on.

See Also

TFirstButton (see page 1913)

TNextButton (see page 1943)

TPriorButton (see page 1949)

TLastButton (see page 1937)

TNextPageButton (see page 1945)

TPriorPageButton (see page 1951)

TInsertButton (see page 1935)

TDeleteButton (see page 1867)

TUndoButton (see page 2055)

TPostButton (see page 1947)

IWebContent

TDataNavigator (see page 1855)

37.1.13.1 TDataSetButton Members

The following tables list the members exposed by TDataSetButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.
V	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.
V	Notification (see page 2194)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
R	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
V	Create (see page 2059)	Creates and initializes an instance of TWebButton (see page 2058).
V	Create (see page 2193)	Creates and initializes an instance of TXMLDisplayReferenceButton (see page 2191).
V	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.
	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.13.2 TDataSetButton Fields

The fields of the TDataSetButton class are listed here.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

37.1.13.2.1 TDataSetButton.DefaultCaption Field

Pascal

```
DefaultCaption: string;
```

C++

```
UnicodeString DefaultCaption;
```

Description

This is DefaultCaption, a member of class TDataSetButton.

37.1.13.2.2 TDataSetButton.XMLMethodName Field

Pascal

```
XMLMethodName: string;
```

C++

```
UnicodeString XMLMethodName;
```

Description

This is XMLMethodName, a member of class TDataSetButton.

37.1.13.3 TDataSetButton Methods

The methods of the TDataSetButton class are listed here.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.
V	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.

37.1.13.3.1 TDataSetButton.AddElements Method

Pascal

```
procedure AddElements(AddIntf: IAddScriptElements);
```

C++

```
__fastcall AddElements(IAddScriptElements AddIntf);
```

Description

This is AddElements, a member of class TDataSetButton.

37.1.13.3.2 TDataSetButton.GetCaption Method

Returns the value of the Caption property.

Pascal

```
function GetCaption: string; override;
```

C++

```
virtual __fastcall UnicodeString GetCaption();
```

Description

GetCaption is the protected read implementation for the Caption property.

See Also

Caption (see page 2062)

SetCaption (see page 2061)

37.1.13.3.3 TDataSetButton.GetMethodName Method

Pascal

```
function GetMethodName: string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetMethodName();
```

Description

This is GetMethodName, a member of class TDataSetButton.

37.1.13.3.4 TDataSetButton.GetSubComponents Method

Pascal

```
function GetSubComponents: TObject;
```

C++

```
__fastcall TObject * GetSubComponents();
```

Description

This is GetSubComponents, a member of class TDataSetButton.

37.1.13.3.5 TDataSetButton.ImplAddElements Method

Pascal

```
procedure ImplAddElements(AddIntf: IAddScriptElements); virtual;
```

C++

```
virtual __fastcall ImplAddElements(IAddScriptElements AddIntf);
```

Description

This is ImplAddElements, a member of class TDataSetButton.

37.1.13.3.6 TDataSetButton.ImplContent Method

Provides the underlying implementation for the Content method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content method calls ImplContent to generate the HTML that defines the button control. TWebButton introduces ImplContent as an abstract or, in C++ terminology, pure virtual method. TWebButton descendants must override ImplContent to generate the definition of their particular button control.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the Web button should format the HTML control it generates. It implements the ILayoutWebContent interface.

ImplContent returns a string of HTML defining the button control.

See Also

Content (see page 2060)

ILayoutWebContent

37.1.13.4 TDataSetButton Properties

The properties of the TDataSetButton class are listed here.

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.
	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.13.4.1 TDataSetButton.Caption Property

Specifies the caption that appears on the button.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the caption that appears on the generated HTML button. The value of Caption is used as the value of a VALUE parameter on the HTML INPUT tag generated to implement the button.

See Also

Custom (see page 2062)

37.1.13.4.2 TDataSetButton.Custom Property

Represents a tag option and its value in the generated INPUT tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

Use Custom to represent one or more options that qualify the INPUT tag generated by the Web button. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to add an ALIGN option that aligns the caption and a TABINDEX tag to control tab order, set Custom to

```
ALIGN=TOP TABINDEX=2
```

See Also

Caption (see page 2062)

Style (see page 2063)

37.1.13.4.3 TDataSetButton.Style Property

Adds style attributes to the generated button control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to add a STYLE option that qualifies the INPUT tag generated by the Web button. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated INPUT tag as the value of a STYLE option.

See Also

Custom (see page 2062)

StyleRule (see page 2063)

37.1.13.4.4 TDataSetButton.StyleRule Property

Applies a style in the page producer's style sheet to the button.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated button control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the button's Style property.

See Also

Style (see page 2063)

Styles (see page 2272)

StylesFile (see page 2273)

Styles

StylesFile

37.1.13.4.5 TDataSetButton.XMLComponent Property

Associates the button with the component that generates the display control it influences.

Pascal

```
property XMLComponent: TComponent;
```

C++

```
__property TComponent * XMLComponent;
```

Description

Set XMLComponent to specify the Web item that generates a display control that the button influences.

Note: Typically, descendants of TXMLDisplayReferenceButton are used by a descendant of TXMLDisplayReferenceGroup, which has its own XMLComponent property. If the button influences this same component, you need not set the XMLComponent property. Instead, setting the XMLUseParent property to true (the default) sets XMLComponent to the value of the parent's XMLComponent property.

See Also

XMLUseParent (see page 2196)

37.1.13.4.6 TDataSetButton.XMLUseParent Property

Specifies whether the button is associated with its parent's display object.

Pascal

```
property XMLUseParent: Boolean;
```

C++

```
__property Boolean XMLUseParent;
```

Description

Use XMLUseParent to indicate how the dataset button determines the component that it influences. When XMLUseParent is false, the XMLComponent property must be set to indicate the display component. When XMLUseParent is true, the button influences the component specified by its parent's XMLComponent property.

The button's parent is the Web item that uses the button to assemble its own HTML content (typically a TDataNavigator or TQueryButtons object). If the parent does not have an associated display component, setting XMLUseParent to true sets the XMLComponent property to nil (Delphi) or NULL (C++).

See Also

XMLComponent (see page 2195)

XMLComponent (see page 2199)

37.1.14 TDeleteButton Class

TDeleteButton generates an HTML control that deletes a record from an XML data packet.

Class Hierarchy

Pascal

```
TDeleteButton = class(TDataSetButton);
```

C++

```
class TDeleteButton : public TDataSetButton;
```

File

MidItems

Description

Add TDeleteButton as the child of a TDataNavigator (see page 1855) component (or a layout group that is a child of a TDataNavigator (see page 1855) component) to enable the generated HTML form to delete records. The resulting HTML button's action adds the record deletion to the emerging delta packet, which can then be submitted using an apply updates button.

Use the properties of TDeleteButton to customize the appearance of the button and associate it with a Web item. The associated Web item displays the records that the delete button can delete.

See Also

TXMLBroker

TDataNavigator (see page 1855)

TLayoutGroup (see page 1939)

TApplyUpdatesButton (see page 1798)

37.1.14.1 TDeleteButton Members

The following tables list the members exposed by TDeleteButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.
V	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 2194)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
R	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
V	Create (see page 1869)	Creates and initializes an instance of TDeleteButton (see page 1867).
V	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.
	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.14.2 TDeleteButton.Create Constructor

Creates and initializes an instance of TDeleteButton (see page 1867).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TDeleteButton(TComponent * AOwner);
```

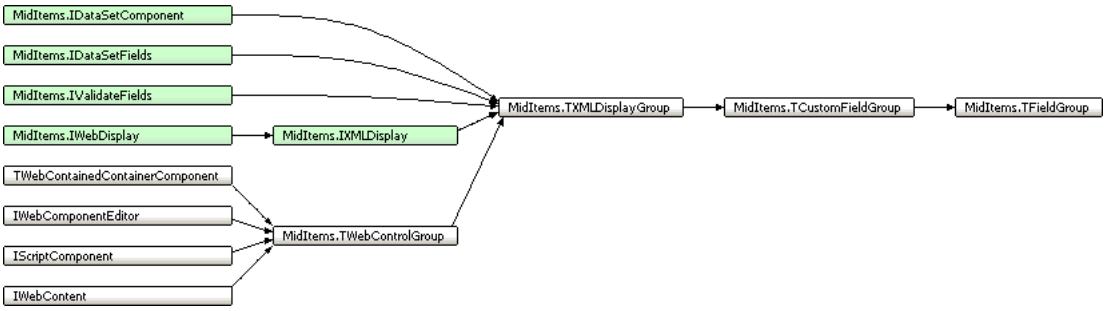
Description

Call Create to instantiate TDeleteButton (see page 1867) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.15 TFieldGroup Class

TFieldGroup arranges database information on an HTML form as a set of labeled fields.

Class Hierarchy



Pascal

```
TFieldGroup = class(TCustomFieldGroup);
```

C++

```
class TFieldGroup : public TCustomFieldGroup;
```

File

MidItems

Description

Add TFieldGroup to a data form component (TDataForm (see page 1847)) or a layout group (TLayoutGroup (see page 1939)) that is a child for a data form. TFieldGroup generates the HTML to display information from an XML data packet in labeled fields.

TFieldGroup maintains a list of Web items that generate the individual display fields. These Web items are restricted to the following classes: TFieldText (see page 1899), TFieldTextArea (see page 1905), TFieldSelectOptions (see page 1884), TFieldRadioGroup (see page 1874), and TFieldStatus (see page 1894). You can bind the field group to an XML data packet or dataset field within an XML data packet. The Web items can then use that binding to determine the source of the data they display.

You can also use the properties of TFieldGroup to affect the overall appearance of the set of fields.

See Also

TDataForm (see page 1847)

TLayoutGroup (see page 1939)

TFieldRadioGroup (see page 1874)

TFieldStatus (see page 1894)

TFieldSelectOptions (see page 1884)

TFieldText (see page 1899)

TFieldTextArea (see page 1905)

Using the Web Page Editor

37.1.15.1 TFieldGroup Members

The following tables list the members exposed by TFieldGroup.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	CreateDefaultFields (see page 2179)	This is CreateDefaultFields, a member of class TXMLDisplayGroup.
	FindField (see page 2180)	This is FindField, a member of class TXMLDisplayGroup.
	GetDataSet (see page 2180)	Returns the value of the DataSet property.
	GetDataSetField (see page 2180)	This is GetDataSetField, a member of class TXMLDisplayGroup.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
	GetEnableValidateFields (see page 2180)	This is GetEnableValidateFields, a member of class TXMLDisplayGroup.
V	GetFieldsList (see page 2181)	This is GetFieldsList, a member of class TXMLDisplayGroup.
	GetIsMultipleRecordView (see page 2181)	This is GetIsMultipleRecordView, a member of class TXMLDisplayGroup.
V	GetLayoutAttributes (see page 1828)	Returns an object that specifies how the generated group of controls should look.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
	GetVisibleFields (see page 2181)	This is GetVisibleFields, a member of class TXMLDisplayGroup.
	GetXMLBroker (see page 2181)	Returns the value of the XMLBroker property.
	GetXMLDataSet (see page 2182)	Returns the value of the XMLDataSet property.
	GetXMLDisplayName (see page 2182)	This is GetXMLDisplayName, a member of class TXMLDisplayGroup.
	GetXMLRowSetName (see page 2182)	This is GetXMLRowSetName, a member of class TXMLDisplayGroup.
	HasStatusField (see page 2182)	This is HasStatusField, a member of class TXMLDisplayGroup.
V	ImplAddElements (see page 1872)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 1872)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1828)	Provides the underlying implementation for the Content method.
V	ImplContent (see page 2070)	Provides the underlying implementation for the Content (see page 2068) method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	ImplGetSubComponents (see page 2183)	Provides the underlying implementation for the GetSubComponents method.
V	ImplIsMultipleRecordView (see page 2183)	This is ImplIsMultipleRecordView, a member of class TXMLDisplayGroup.
	IsFieldInUse (see page 2184)	This is IsFieldInUse, a member of class TXMLDisplayGroup.

	Notification (see page 2184)	Forwards notification messages to all owned components.
	SetDataSetField (see page 2184)	This is SetDataSetField, a member of class TXMLDisplayGroup.
	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.
	SetEnableValidateFields (see page 2185)	This is SetEnableValidateFields, a member of class TXMLDisplayGroup.
	SetXMLBroker (see page 2185)	This is SetXMLBroker, a member of class TXMLDisplayGroup.
	ValidateDataSet (see page 2185)	This is ValidateDataSet, a member of class TXMLDisplayGroup.
	ValidateFields (see page 2185)	This is ValidateFields, a member of class TXMLDisplayGroup.
	XMLDataChange (see page 2185)	This is XMLDataChange, a member of class TXMLDisplayGroup.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.
	EnableValidateFields (see page 2186)	This is EnableValidateFields, a member of class TXMLDisplayGroup.
	XMLDataSet (see page 2187)	Indicates the TXMLDataSet object that represents the dataset to which the Web item is bound.

Public Methods

	Name	Description
	Create (see page 1827)	Creates and initializes an instance of TCustomFieldGroup (see page 1824).
	Create (see page 2065)	Creates and initializes an instance of TWebControlGroup (see page 2064).
	Create (see page 2178)	Creates and initializes an instance of TXMLDisplayGroup (see page 2176).
	Destroy (see page 1827)	Frees an instance of TCustomFieldGroup (see page 1824).
	Destroy (see page 2066)	Frees an instance of TWebControlGroup (see page 2064).
	Destroy (see page 2178)	Frees an instance of TXMLDisplayGroup (see page 2176).
	GetDataSet (see page 2201)	Returns the value of the DataSet (see page 2202) property.
	GetEnableValidateFields (see page 2225)	This is GetEnableValidateFields, a member of class IValidateFields.
	GetFieldsList (see page 2207)	This is GetFieldsList, a member of class IDataSetFields.
	GetIsMultipleRecordView (see page 2227)	This is GetIsMultipleRecordView, a member of class IWebDisplay.
	GetIsMultipleRecordView (see page 2229)	This is GetIsMultipleRecordView, a member of class IXMLDisplay.
	GetVisibleFields (see page 2207)	This is GetVisibleFields, a member of class IDataSetFields.
	GetXMLBroker (see page 2201)	Returns the value of the XMLBroker (see page 2203) property.
	GetXMLDataSet (see page 2202)	Returns the value of the XMLDataSet (see page 2203) property.
	GetXMLDisplayName (see page 2230)	This is GetXMLDisplayName, a member of class IXMLDisplay.
	GetXMLRowSetName (see page 2230)	This is GetXMLRowSetName, a member of class IXMLDisplay.
	HasStatusField (see page 2207)	This is HasStatusField, a member of class IDataSetFields.
	IsFieldInUse (see page 2207)	This is IsFieldInUse, a member of class IDataSetFields.
	SetEnableValidateFields (see page 2225)	This is SetEnableValidateFields, a member of class IValidateFields.
	ValidateDataSet (see page 2207)	This is ValidateDataSet, a member of class IDataSetFields.
	ValidateFields (see page 2225)	This is ValidateFields, a member of class IValidateFields.

Public Properties

	Name	Description
	DataSet (see page 2202)	Provides access to an interface for working with the associated dataset.
	EnableValidateFields (see page 2226)	This is EnableValidateFields, a member of class IValidateFields.
	IsMultipleRecordView (see page 2227)	
	IsMultipleRecordView (see page 2230)	
	VisibleFields (see page 2208)	This is VisibleFields, a member of class IDataSetFields.
	VisibleFields (see page 2186)	Indicates the controls displayed by the HTML that the XML display group generates.
	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.
	XMLBroker (see page 2203)	Indicates the XML broker that fetches the data packets to which the Web item is bound.
	XMLDataSet (see page 2203)	Indicates the TXMLDataSet (see page 2164) object that represents the dataset to which the Web item is bound.
	XMLDisplayName (see page 2231)	This is XMLDisplayName, a member of class IXMLDisplay.

	XMLRowSetName (see page 2231)	This is XMLRowSetName, a member of class IXMLDisplay.

Published Properties

	Name	Description
	Custom (see page 1873)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1873)	Adds style attributes to the generated set of fields.
	StyleRule (see page 1874)	Applies a style in the page producer's style sheet to the table.
	XMLBroker (see page 2187)	Binds the XML display group to an XML broker that represents a dataset.
	XMLDataSetField (see page 2188)	Binds the XML display group to a nested dataset from the XML data packet.

37.1.15.2 TFieldGroup Methods

The methods of the TFieldGroup class are listed here.

Protected Methods

	Name	Description
	ImplAddElements (see page 1872)	Provides the underlying implementation for the AddElements method.
	ImplCanAddClass (see page 1872)	Provides the underlying implementation for the CanAddClass method.

37.1.15.2.1 TFieldGroup.ImplAddElements Method

Provides the underlying implementation for the AddElements method.

Pascal

```
procedure ImplAddElements(AddIntf: IAddScriptElements); override;
```

C++

```
virtual __fastcall ImplAddElements(IAddScriptElements AddIntf);
```

Description

The AddElements method calls ImplAddElements to add any script declarations by calling the AddIntf interface. These declarations allow the generated display control to call script methods in response to user events by ensuring that the required methods are declared.

As implemented in TWebControlGroup, ImplAddElements does nothing. TWebControlGroup descendants override ImplAddElements to add any required declarations.

See Also

AddElements (see page 2066)

37.1.15.2.2 TFieldGroup.ImplCanAddClass Method

Provides the underlying implementation for the CanAddClass method.

Pascal

```
function ImplCanAddClass(AParent: TComponent; AClass: TClass): Boolean; override;
```

C++

```
virtual __fastcall Boolean ImplCanAddClass(TComponent * AParent, TClass AClass);
```

Description

The CanAddClass method calls ImpleCanAddClass to determine whether the TWebControlGroup descendant can add a component of type AClass to its WebComponents property list. AParent is the TWebControlGroup instance to which the

component will be added.

In TWebControlGroup, ImplCanAddClass always returns true. TWebControlGroup descendants override ImplCanAddClass to indicate what types of Web Items they can use when generating their HTML content.

See Also

CanAddClass (see page 2067)

WebFieldControls (see page 2072)

37.1.15.3 TFieldGroup Properties

The properties of the TFieldGroup class are listed here.

Published Properties

	Name	Description
	Custom (see page 1873)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1873)	Adds style attributes to the generated set of fields.
	StyleRule (see page 1874)	Applies a style in the page producer's style sheet to the table.

37.1.15.3.1 TFieldGroup.Custom Property

Represents a tag option and its value in the generated TABLE tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TCustomFieldGroup generates an HTML table to manage the layout of the HTML controls that its Web items generate. Use Custom to add options that qualify the TABLE tag. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to surround the set of fields with a border and give them a blue background, set Custom to

```
BORDER=2 BGCOLOR=BLUE
```

See Also

Style (see page 1829)

StyleRule (see page 1830)

37.1.15.3.2 TFieldGroup.Style Property

Adds style attributes to the generated set of fields.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

TCustomFieldGroup generates an HTML table to lay out the fields that its Web items generate. Use Style to add a STYLE option that qualifies the TABLE tag. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated TABLE tag as the value of a STYLE option.

See Also

Custom (see page 1829)

StyleRule (see page 1830)

37.1.15.3.3 TFieldGroup.StyleRule Property

Applies a style in the page producer's style sheet to the table.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated set of fields. The value of StyleRule specifies one of the user-defined style names defined in the InternetExpress page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the Style property.

See Also

Style (see page 1829)

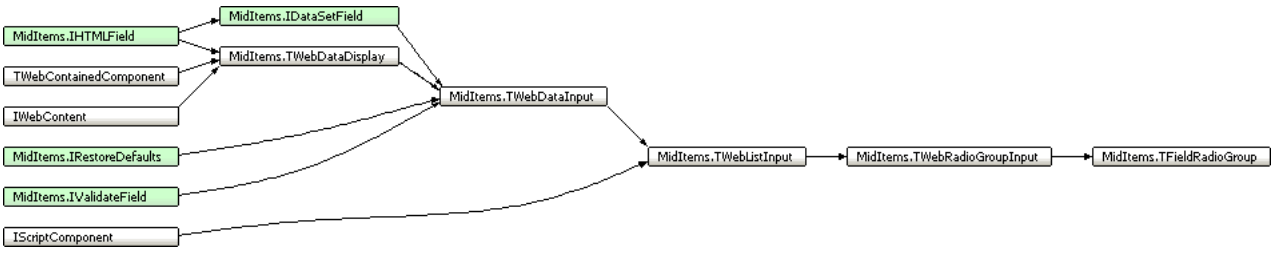
Styles (see page 2272)

StylesFile (see page 2273)

37.1.16 TFieldRadioGroup Class

TFieldRadioGroup creates an HTML radio group control to represent a field in an XML data packet.

Class Hierarchy



Pascal

```
TFieldRadioGroup = class(TWebRadioGroupInput);
```

C++

```
class TFieldRadioGroup : public TWebRadioGroupInput;
```

File

MidItems

Description

Add TFieldRadioGroup as the child of a TFieldGroup (see page 1869) component to add a radio group control that represents a field from the field group's associated dataset.

Use the properties and methods of TFieldRadioGroup to

Specify the field whose value the radio group displays.

Specify the possible values for the field from which the user can choose.

Specify a caption that describes the field.

Customize the appearance of the radio group.

Fit the radio group into the tab order of the generated HTML form.

See Also

TFieldGroup (see page 1869)

TXMLDisplayGroup (see page 2176)

37.1.16.1 TFieldRadioGroup Members

The following tables list the members exposed by TFieldRadioGroup.

Protected Methods

	Name	Description
V	AddAttributes (see page 2125)	This is AddAttributes, a member of class TWebRadioGroupInput.
	AddElements (see page 2117)	This is AddElements, a member of class TWebListInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
A	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
V	ControlContent (see page 2117)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
V	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
V	EventContent (see page 2126)	This is EventContent, a member of class TWebRadioGroupInput.
V	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
A	FormatInputs (see page 2117)	This is FormatInputs, a member of class TWebListInput.
V	FormatInputs (see page 2126)	This is FormatInputs, a member of class TWebRadioGroupInput.
V	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
V	GetCaption (see page 2090)	Returns the value of the Caption property.
V	GetCheckIndex (see page 2126)	This is GetCheckIndex, a member of class TWebRadioGroupInput.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
V	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.

	GetSubComponents (see page 2118)	This is GetSubComponents, a member of class TWebListInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplAddElements (see page 2118)	This is ImplAddElements, a member of class TWebListInput.
V	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
V	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
V	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
V	ImplGetRowSetFieldAttributes (see page 2126)	This is ImplGetRowSetFieldAttributes, a member of class TWebRadioGroupInput.
V	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
V	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
V	Notification (see page 2118)	Forwards notification messages to all owned components.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
V	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
V	RestoreFieldDefaults (see page 2126)	This is RestoreFieldDefaults, a member of class TWebRadioGroupInput.
V	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
V	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetDataSet (see page 2119)	This is SetDataSet, a member of class TWebListInput.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetItems (see page 2119)	This is SetItems, a member of class TWebListInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	SetValues (see page 2119)	This is SetValues, a member of class TWebListInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
V	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
V	Create (see page 2116)	Creates and initializes an instance of TWebListInput (see page 2113).
V	Create (see page 2125)	Creates and initializes an instance of TWebRadioGroupInput (see page 2122).
V	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
V	Destroy (see page 2116)	Frees an instance of TWebListInput (see page 2113).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
V	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
V	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

Published Properties

	Name	Description
	Caption (see page 1878)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1878)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1878)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1879)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 1879)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayColumns (see page 1880)	Specifies the number of radio buttons that appear in each row of the control.
	DisplayWidth (see page 1880)	Specifies the width of the radio group, in characters.
	FieldName (see page 1880)	Specifies the name of the field whose value the generated control represents.
	Items (see page 1881)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 1881)	Indicates which field in DataSet supplies the possible selections for the generated control.
	ReadOnly (see page 1882)	Indicates whether end users can use the generated control to set values.
	Style (see page 1882)	Adds style attributes to the generated display control.
	StyleRule (see page 1883)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1883)	Determines the position of the generated control in the HTML form's tab order.
	Values (see page 1883)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 1884)	Indicates which field in DataSet supplies the possible values for the generated control.

37.1.16.2 TFieldRadioGroup Properties

The properties of the TFieldRadioGroup class are listed here.

Published Properties

	Name	Description
	Caption (see page 1878)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1878)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1878)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1879)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 1879)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayColumns (see page 1880)	Specifies the number of radio buttons that appear in each row of the control.
	DisplayWidth (see page 1880)	Specifies the width of the radio group, in characters.
	FieldName (see page 1880)	Specifies the name of the field whose value the generated control represents.
	Items (see page 1881)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 1881)	Indicates which field in DataSet supplies the possible selections for the generated control.
	ReadOnly (see page 1882)	Indicates whether end users can use the generated control to set values.
	Style (see page 1882)	Adds style attributes to the generated display control.
	StyleRule (see page 1883)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1883)	Determines the position of the generated control in the HTML form's tab order.
	Values (see page 1883)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 1884)	Indicates which field in DataSet supplies the possible values for the generated control.

37.1.16.2.1 TFieldRadioGroup.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.16.2.2 TFieldRadioGroup.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.16.2.3 TFieldRadioGroup.CaptionPosition Property

Specifies where the label appears relative to the generated display control.

Pascal

```
property CaptionPosition: TCaptionPosition;
```

C++

```
__property TCaptionPosition CaptionPosition;
```

Description

Use CaptionPosition to indicate where the label should appear relative to the control that displays information.

Note: CaptionPosition has no effect on column controls. The position of column labels is always in the table header.

See Also

CaptionAttributes (see page 2084)

Caption (see page 2083)

37.1.16.2.4 TFieldRadioGroup.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.16.2.5 TFieldRadioGroup.DataSet Property

Specifies a dataset that supplies the possible values listed in the generated control.

Pascal

```
property DataSet: TDataSet;
```

C++

```
__property TDataSet DataSet;
```

Description

Use DataSet to specify a dataset from which the component can extract the selections that appear to the user and the underlying values they represent. The ItemsField property specifies the field in this dataset that lists the selections that appear in the generated control. The ValuesField property specifies the field in this dataset that lists the corresponding values that are assigned to the control.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

ItemsField (see page 2120)

ValuesField (see page 2121)

Items (see page 2120)

Values (see page 2121)

37.1.16.2.6 TFieldRadioGroup.DisplayColumns Property

Specifies the number of radio buttons that appear in each row of the control.

Pascal

```
property DisplayColumns: Integer;
```

C++

```
__property int DisplayColumns;
```

Description

Use DisplayColumns to specify how many radio buttons can appear in a single line of the radio group. If DisplayColumns is less than the number of items, the radio group displays its buttons in multiple rows.

See Also

ItemsField (see page 2120)

Items (see page 2120)

DisplayWidth (see page 2127)

37.1.16.2.7 TFieldRadioGroup.DisplayWidth Property

Specifies the width of the radio group, in characters.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

Description

Use DisplayWidth to limit the width of the generated control. TWebRadioGroup generates an INPUT tag of type RADIO to implement the radio group. DisplayWidth is added as a WIDTH option.

See Also

DisplayColumns (see page 2127)

Custom (see page 2084)

37.1.16.2.8 TFieldRadioGroup.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.16.2.9 TFieldRadioGroup.Items Property

Lists the possible selections that appear in the generated control.

Pascal

```
property Items: TStrings;
```

C++

```
__property TStrings Items;
```

Description

Use Items to supply a list of possible selections to the generated control. The end-user can choose from these selections to assign a value to the generated control. The value that is assigned comes from the corresponding entry in the Values property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items and Values properties, use the DataSet, ItemsField, and ValuesField properties.

See Also

ItemsField (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.16.2.10 TFieldRadioGroup.ItemsField Property

Indicates which field in DataSet supplies the possible selections for the generated control.

Pascal

```
property ItemsField: string;
```

C++

```
__property UnicodeString ItemsField;
```

Description

Use ItemsField to specify which field in the dataset specified by the DataSet property contains selections that appear in the generated control. The end-user can choose from these selections to assign a value to the generated control. The value that is

assigned comes from the field specified by the ValuesField property.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.16.2.11 TFieldRadioGroup.ReadOnly Property

Indicates whether end users can use the generated control to set values.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use ReadOnly to indicate whether users can change the value displayed by the radio group. TWebRadioGroup generates an INPUT tag of type RADIO to implement the radio group. ReadOnly is added as a READONLY option.

See Also

DisplayWidth (see page 2127)

Custom (see page 2084)

37.1.16.2.12 TFieldRadioGroup.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.16.2.13 TFieldRadioGroup.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.16.2.14 TFieldRadioGroup.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.16.2.15 TFieldRadioGroup.Values Property

Lists the possible values that can be assigned to the control.

Pascal

```
property Values: TStrings;
```

C++

```
__property TStrings Values;
```

Description

Use Values to supply a list of possible values for the generated control. When the end-user chooses one of the selections listed by the Items property, the control is assigned the corresponding value in the Values property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items and Values properties, use the DataSet, ItemsField, and ValuesField properties.

See Also

Items (see page 2120)

ItemsField (see page 2120)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.16.2.16 TFieldRadioGroup.ValuesField Property

Indicates which field in DataSet supplies the possible values for the generated control.

Pascal

```
property ValuesField: string;
```

C++

```
__property UnicodeString ValuesField;
```

Description

Use ValuesField to specify which field in the dataset specified by the DataSet property contains values that can be assigned to the generated control. When the end-user selects the corresponding value form the ItemsField field, the control gets the value in the ValuesField field.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

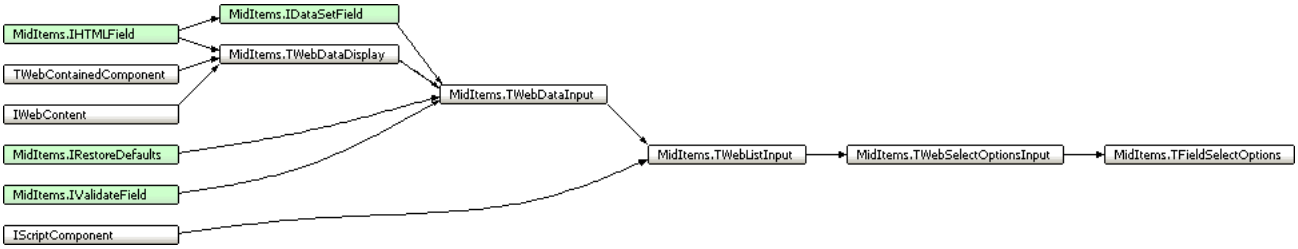
DataSet (see page 2119)

ItemsField (see page 2120)

37.1.17 TFieldSelectOptions Class

TFieldSelectOptions creates an HTML list control to represent a field in an XML data packet.

Class Hierarchy



Pascal

```
TFieldSelectOptions = class(TWebSelectOptionsInput);
```

C++

```
class TFieldSelectOptions : public TWebSelectOptionsInput;
```

File

MidItems

Description

Add TFieldSelectOptions as the child of a TFieldGroup (see page 1869) component to add a list control that represents a field from the field group's associated dataset.

Use the properties and methods of TFieldSelectOptions to

Specify the field whose value the list box displays.

Specify the possible values for the field from which the user can choose.

Specify a caption that describes the field.

Customize the appearance of the list box.

Fit the list box into the tab order of the generated HTML form.

See Also

TFieldGroup (see page 1869)

TXMLDisplayGroup (see page 2176)

37.1.17.1 TFieldSelectOptions Members

The following tables list the members exposed by TFieldSelectOptions.

Protected Methods

	Name	Description
	AddAttributes (see page 2131)	This is AddAttributes, a member of class TWebSelectOptionsInput.
	AddElements (see page 2117)	This is AddElements, a member of class TWebListInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
	ControlContent (see page 2117)	Returns the HTML that defines the display control (without its caption).
	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
	EventContent (see page 2132)	This is EventContent, a member of class TWebSelectOptionsInput.

	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
	FormatInputs (see page 2117)	This is FormatInputs, a member of class TWebListInput.
	FormatInputs (see page 2132)	This is FormatInputs, a member of class TWebSelectOptionsInput.
	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetSelectIndex (see page 2132)	This is GetSelectIndex, a member of class TWebSelectOptionsInput.
	GetSubComponents (see page 2118)	This is GetSubComponents, a member of class TWebListInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
	ImplAddElements (see page 2118)	This is ImplAddElements, a member of class TWebListInput.
	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	Notification (see page 2118)	Forwards notification messages to all owned components.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetDataSet (see page 2119)	This is SetDataSet, a member of class TWebListInput.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetItems (see page 2119)	This is SetItems, a member of class TWebListInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	SetValues (see page 2119)	This is SetValues, a member of class TWebListInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
	Create (see page 2116)	Creates and initializes an instance of TWebListInput (see page 2113).
	Create (see page 2131)	Creates and initializes an instance of TWebSelectOptionsInput (see page 2128).
	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).

	Destroy (see page 2116)	Frees an instance of TWebListInput (see page 2113).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

Published Properties

	Name	Description
	Caption (see page 1888)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1888)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1888)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1889)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 1889)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayRows (see page 1890)	Specifies the number of rows that appear in the generated list box.
	FieldName (see page 1890)	Specifies the name of the field whose value the generated control represents.
	Items (see page 1891)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 1891)	Indicates which field in DataSet supplies the possible selections for the generated control.
	Style (see page 1891)	Adds style attributes to the generated display control.
	StyleRule (see page 1892)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1892)	Determines the position of the generated control in the HTML form's tab order.
	Values (see page 1893)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 1893)	Indicates which field in DataSet supplies the possible values for the generated control.

37.1.17.2 TFieldSelectOptions Properties

The properties of the TFieldSelectOptions class are listed here.

Published Properties

	Name	Description
	Caption (see page 1888)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1888)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1888)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1889)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 1889)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayRows (see page 1890)	Specifies the number of rows that appear in the generated list box.
	FieldName (see page 1890)	Specifies the name of the field whose value the generated control represents.
	Items (see page 1891)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 1891)	Indicates which field in DataSet supplies the possible selections for the generated control.
	Style (see page 1891)	Adds style attributes to the generated display control.

	StyleRule (see page 1892)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1892)	Determines the position of the generated control in the HTML form's tab order.
	Values (see page 1893)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 1893)	Indicates which field in DataSet supplies the possible values for the generated control.

37.1.17.2.1 TFieldSelectOptions.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.17.2.2 TFieldSelectOptions.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.17.2.3 TFieldSelectOptions.CaptionPosition Property

Specifies where the label appears relative to the generated display control.

Pascal

```
property CaptionPosition: TCaptionPosition;
```

C++

```
__property TCaptionPosition CaptionPosition;
```

Description

Use CaptionPosition to indicate where the label should appear relative to the control that displays information.

Note: CaptionPosition has no effect on column controls. The position of column labels is always in the table header.

See Also

CaptionAttributes (see page 2084)

Caption (see page 2083)

37.1.17.2.4 TFieldSelectOptions.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.17.2.5 TFieldSelectOptions.DataSet Property

Specifies a dataset that supplies the possible values listed in the generated control.

Pascal

```
property DataSet: TDataSet;
```

C++

```
__property TDataSet DataSet;
```

Description

Use DataSet to specify a dataset from which the component can extract the selections that appear to the user and the underlying values they represent. The ItemsField property specifies the field in this dataset that lists the selections that appear in the generated control. The ValuesField property specifies the field in this dataset that lists the corresponding values that are assigned to the control.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections

that appear to the user and the underlying values they represent.

See Also

ItemsField (see page 2120)

ValuesField (see page 2121)

Items (see page 2120)

Values (see page 2121)

37.1.17.2.6 TFieldSelectOptions.DisplayRows Property

Specifies the number of rows that appear in the generated list box.

Pascal

```
property DisplayRows: Integer;
```

C++

```
__property int DisplayRows;
```

Description

Use DisplayRows to specify how many rows appear in the generated list control. If DisplayRows is less than the number of items, the generated list displays only DisplayRows options at one time, but can scroll to display the hidden items. If DisplayRows is 1, the generated control becomes a drop-down list box.

See Also

ItemsField (see page 2120)

Items (see page 2120)

37.1.17.2.7 TFieldSelectOptions.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.17.2.8 TFieldSelectOptions.Items Property

Lists the possible selections that appear in the generated control.

Pascal

```
property Items: TStrings;
```

C++

```
__property TStrings Items;
```

Description

Use Items to supply a list of possible selections to the generated control. The end-user can choose from these selections to assign a value to the generated control. The value that is assigned comes from the corresponding entry in the Values property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items and Values properties, use the DataSet, ItemsField, and ValuesField properties.

See Also

ItemsField (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.17.2.9 TFieldSelectOptions.ItemsField Property

Indicates which field in DataSet supplies the possible selections for the generated control.

Pascal

```
property ItemsField: string;
```

C++

```
__property UnicodeString ItemsField;
```

Description

Use ItemsField to specify which field in the dataset specified by the DataSet property contains selections that appear in the generated control. The end-user can choose from these selections to assign a value to the generated control. The value that is assigned comes from the field specified by the ValuesField property.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.17.2.10 TFieldSelectOptions.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.17.2.11 TFieldSelectOptions.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.17.2.12 TFieldSelectOptions.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.17.2.13 TFieldSelectOptions.Values Property

Lists the possible values that can be assigned to the control.

Pascal

```
property Values: TStrings;
```

C++

```
__property TStrings Values;
```

Description

Use Values to supply a list of possible values for the generated control. When the end-user chooses one of the selections listed by the Items property, the control is assigned the corresponding value in the Values property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items and Values properties, use the DataSet, ItemsField, and ValuesField properties.

See Also

Items (see page 2120)

ItemsField (see page 2120)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.17.2.14 TFieldSelectOptions.ValuesField Property

Indicates which field in DataSet supplies the possible values for the generated control.

Pascal

```
property ValuesField: string;
```

C++

```
__property UnicodeString ValuesField;
```

Description

Use ValuesField to specify which field in the dataset specified by the DataSet property contains values that can be assigned to the generated control. When the end-user selects the corresponding value form the ItemsField field, the control gets the value in the ValuesField field.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

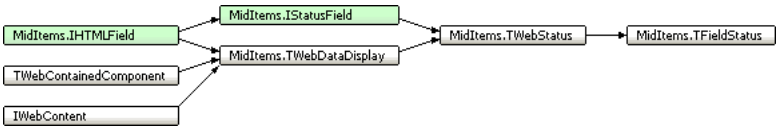
DataSet (see page 2119)

ItemsField (see page 2120)

37.1.18 TFieldStatus Class

TFieldStatus generates an HTML control that indicates whether a record in an XML data packet has been modified by the end user.

Class Hierarchy



Pascal

```
TFieldStatus = class(TWebStatus);
```

C++

```
class TFieldStatus : public TWebStatus;
```

File

MidItems

Description

Add TFieldStatus as the child of a TFieldGroup (see page 1869) component to provide feedback about whether the current record has been modified. The generated control displays an M if the user has edited the record, an I if the user has inserted the record, and blank if the user has made no changes to the record.

Use the properties and methods of TFieldStatus to

Specify a caption that labels the status control.

Customize the appearance of the control.

Fit the control into the tab order of the generated HTML form.

See Also

TFieldGroup (see page 1869)

TXMLDisplayGroup (see page 2176)

37.1.18.1 TFieldStatus Members

The following tables list the members exposed by TFieldStatus.

Protected Methods

	Name	Description
	Content (see page 2076)	Returns the HTML that defines the display control.
A	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
V	ControlContent (see page 2135)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
V	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
V	GetCaption (see page 2136)	Returns the value of the Caption property.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
V	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetHTMLControlName (see page 2136)	Returns the name that other components can use to refer to the generated display control.
V	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
V	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.

Protected Properties

	Name	Description
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.

Public Methods

	Name	Description
V	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
V	Create (see page 2135)	Creates and initializes an instance of TWebStatus (see page 2133).
V	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	Identifier (see page 2136)	Returns the name under which record status information is stored in the XML data packet.
V	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
V	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.

Public Properties

	Name	Description
	DisplayWidth (see page 2138)	Specifies the width, in characters, of the generated HTML control.
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	TabIndex (see page 2086)	Determines the position of the generated control in the HTML form's tab order.

Published Properties

	Name	Description
	Caption (see page 1896)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1896)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1897)	Specifies where the label appears relative to the generated display control.

	Custom (see page 1897)	Represents a tag option and its value in the tag for the display control.
	Style (see page 1897)	Adds style attributes to the generated display control.
	StyleRule (see page 1898)	Applies a style in the page producer's style sheet to the generated display control.

37.1.18.2 TFieldStatus Properties

The properties of the TFieldStatus class are listed here.

Published Properties

	Name	Description
	Caption (see page 1896)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1896)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1897)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1897)	Represents a tag option and its value in the tag for the display control.
	Style (see page 1897)	Adds style attributes to the generated display control.
	StyleRule (see page 1898)	Applies a style in the page producer's style sheet to the generated display control.

37.1.18.2.1 TFieldStatus.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.18.2.2 TFieldStatus.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.18.2.3 TFieldStatus.CaptionPosition Property

Specifies where the label appears relative to the generated display control.

Pascal

```
property CaptionPosition: TCaptionPosition;
```

C++

```
__property TCaptionPosition CaptionPosition;
```

Description

Use CaptionPosition to indicate where the label should appear relative to the control that displays information.

Note: CaptionPosition has no effect on column controls. The position of column labels is always in the table header.

See Also

CaptionAttributes (see page 2084)

Caption (see page 2083)

37.1.18.2.4 TFieldStatus.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.18.2.5 TFieldStatus.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.18.2.6 TFieldStatus.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

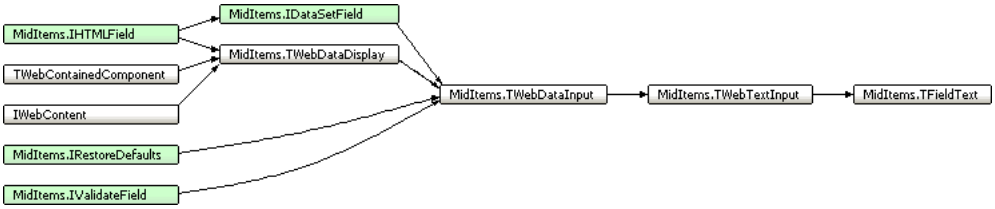
StylesFile

CaptionAttributes (see page 2084)

37.1.19 TFieldText Class

TFieldText creates a single-line HTML input control to represent a field in an XML data packet.

Class Hierarchy



Pascal

```
TFieldText = class(TWebTextInput);
```

C++

```
class TFieldText : public TWebTextInput;
```

File

MidItems

Description

Add TFieldText as the child of a TFieldGroup (see page 1869) component to add a single-line edit control that represents a field from the field group's associated dataset.

Use the properties and methods of TFieldText to

Specify the field whose value the control displays.

Specify a caption that describes the field.

Customize the appearance of the edit control.

Fit the control into the tab order of the generated HTML form.

See Also

TFieldGroup (see page 1869)

TXMLDisplayGroup (see page 2176)

37.1.19.1 TFieldText Members

The following tables list the members exposed by TFieldText.

Protected Methods

	Name	Description
	AddAttributes (see page 2148)	This is AddAttributes, a member of class TWebTextInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
	ControlContent (see page 2148)	Returns the HTML that defines the display control (without its caption).
	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.

	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
	ImplGetRowSetFieldAttributes (see page 2149)	This is ImplGetRowSetFieldAttributes, a member of class TWebTextInput.
	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
	RestoreFieldDefaults (see page 2149)	This is RestoreFieldDefaults, a member of class TWebTextInput.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
	Create (see page 2147)	Creates and initializes an instance of TWebTextInput (see page 2145).
	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.

	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

Published Properties

	Name	Description
	Caption (see page 1901)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1902)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1902)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1902)	Represents a tag option and its value in the tag for the display control.
	DisplayWidth (see page 1903)	Specifies the horizontal size of the generated control.
	FieldName (see page 1903)	Specifies the name of the field whose value the generated control represents.
	ReadOnly (see page 1904)	Indicates whether end users can use the generated control to set values.
	Style (see page 1904)	Adds style attributes to the generated display control.
	StyleRule (see page 1904)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1905)	Determines the position of the generated control in the HTML form's tab order.

37.1.19.2 TFieldText Properties

The properties of the TFieldText class are listed here.

Published Properties

	Name	Description
	Caption (see page 1901)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1902)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1902)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1902)	Represents a tag option and its value in the tag for the display control.
	DisplayWidth (see page 1903)	Specifies the horizontal size of the generated control.
	FieldName (see page 1903)	Specifies the name of the field whose value the generated control represents.
	ReadOnly (see page 1904)	Indicates whether end users can use the generated control to set values.
	Style (see page 1904)	Adds style attributes to the generated display control.
	StyleRule (see page 1904)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1905)	Determines the position of the generated control in the HTML form's tab order.

37.1.19.2.1 TFieldText.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.19.2.2 TFieldText.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.19.2.3 TFieldText.CaptionPosition Property

Specifies where the label appears relative to the generated display control.

Pascal

```
property CaptionPosition: TCaptionPosition;
```

C++

```
__property TCaptionPosition CaptionPosition;
```

Description

Use CaptionPosition to indicate where the label should appear relative to the control that displays information.

Note: CaptionPosition has no effect on column controls. The position of column labels is always in the table header.

See Also

CaptionAttributes (see page 2084)

Caption (see page 2083)

37.1.19.2.4 TFieldText.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.19.2.5 TFieldText.DisplayWidth Property

Specifies the horizontal size of the generated control.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

Description

Use DisplayWidth to indicate the desired width of the generated HTML control. DisplayWidth specifies the number of characters that can be displayed in the control.

Note: DisplayWidth controls the size of the generated control, not the size of the text that users can enter. If the user enters a string longer than DisplayWidth characters, the text scrolls in the control.

37.1.19.2.6 TFieldText.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.19.2.7 TFieldText.ReadOnly Property

Indicates whether end users can use the generated control to set values.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use ReadOnly to indicate whether users can change the value displayed by the generated control.

37.1.19.2.8 TFieldText.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.19.2.9 TFieldText.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The

value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.19.2.10 TFieldText.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

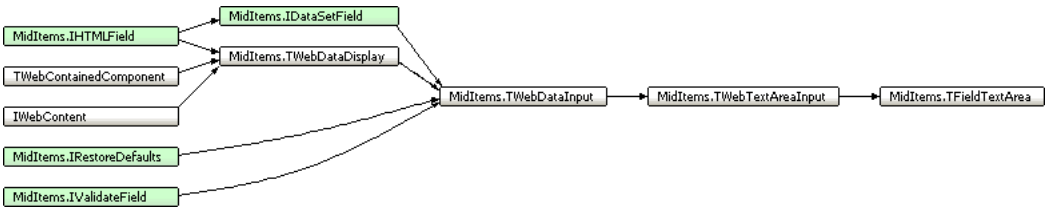
Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.20 TFieldTextArea Class

TFieldTextArea creates a multi-line HTML input control to represent a field in an XML data packet.

Class Hierarchy



Pascal

```
TFieldTextArea = class(TWebTextAreaInput);
```

C++

```
class TFieldTextArea : public TWebTextAreaInput;
```

File

MidItems

Description

Add TFieldTextArea as the child of a TFieldGroup (see page 1869) component to add a multi-line edit control that represents a field from the field group's associated dataset.

Use the properties and methods of TFieldTextArea to

Specify the field whose value the control displays.

Specify a caption that describes the field.

Customize the appearance of the control.

Fit the control into the tab order of the generated HTML form.

See Also

IWebContent

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

TQueryTextArea (see page 2010)

TTextAreaColumn (see page 2041)

37.1.20.1 TFieldTextArea Members

The following tables list the members exposed by TFieldTextArea.

Protected Methods

	Name	Description
	AddAttributes (see page 2142)	This is AddAttributes, a member of class TWebTextAreaInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
	ControlContent (see page 2142)	Returns the HTML that defines the display control (without its caption).
	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
	EventContent (see page 2142)	This is EventContent, a member of class TWebTextAreaInput.
	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.

	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
	ImplGetRowSetFieldAttributes (see page 2143)	This is ImplGetRowSetFieldAttributes, a member of class TWebTextAreaInput.
	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
	RestoreFieldDefaults (see page 2143)	This is RestoreFieldDefaults, a member of class TWebTextAreaInput.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
	Create (see page 2141)	Creates and initializes an instance of TWebTextAreaInput (see page 2138).
	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

Published Properties

	Name	Description
	Caption (see page 1908)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1908)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1909)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1909)	Represents a tag option and its value in the tag for the display control.
	DisplayRows (see page 1910)	Specifies the height of the generated control.
	DisplayWidth (see page 1910)	Specifies the horizontal size of the generated control.
	FieldName (see page 1910)	Specifies the name of the field whose value the generated control represents.
	ReadOnly (see page 1911)	Indicates whether end users can use the generated control to set values.

	Style (see page 1911)	Adds style attributes to the generated display control.
	StyleRule (see page 1911)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1912)	Determines the position of the generated control in the HTML form's tab order.
	Wrap (see page 1912)	Specifies how the generated control treats lines that are longer than the display width.

37.1.20.2 TFieldTextArea Properties

The properties of the TFieldTextArea class are listed here.

Published Properties

	Name	Description
	Caption (see page 1908)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1908)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1909)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1909)	Represents a tag option and its value in the tag for the display control.
	DisplayRows (see page 1910)	Specifies the height of the generated control.
	DisplayWidth (see page 1910)	Specifies the horizontal size of the generated control.
	FieldName (see page 1910)	Specifies the name of the field whose value the generated control represents.
	ReadOnly (see page 1911)	Indicates whether end users can use the generated control to set values.
	Style (see page 1911)	Adds style attributes to the generated display control.
	StyleRule (see page 1911)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1912)	Determines the position of the generated control in the HTML form's tab order.
	Wrap (see page 1912)	Specifies how the generated control treats lines that are longer than the display width.

37.1.20.2.1 TFieldTextArea.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.20.2.2 TFieldTextArea.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.20.2.3 TFieldTextArea.CaptionPosition Property

Specifies where the label appears relative to the generated display control.

Pascal

```
property CaptionPosition: TCaptionPosition;
```

C++

```
__property TCaptionPosition CaptionPosition;
```

Description

Use CaptionPosition to indicate where the label should appear relative to the control that displays information.

Note: CaptionPosition has no effect on column controls. The position of column labels is always in the table header.

See Also

CaptionAttributes (see page 2084)

Caption (see page 2083)

37.1.20.2.4 TFieldTextArea.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.20.2.5 TFieldTextArea.DisplayRows Property

Specifies the height of the generated control.

Pascal

```
property DisplayRows: Integer;
```

C++

```
__property int DisplayRows;
```

Description

Use DisplayRows to indicate the desired height of the generated HTML control. DisplayRows specifies the number of lines of text that can be displayed in the control. This value is assigned to the ROWS option of the generated TEXTAREA tag.

See Also

DisplayWidth (see page 2144)

37.1.20.2.6 TFieldTextArea.DisplayWidth Property

Specifies the horizontal size of the generated control.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

Description

Use DisplayWidth to indicate the desired width of the generated HTML control. DisplayWidth specifies the number of characters that can be displayed in one line of the control. This value is assigned to the COLS option of the generated TEXTAREA tag.

See Also

DisplayRows (see page 2143)

37.1.20.2.7 TFieldTextArea.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a

parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.20.2.8 TFieldTextArea.ReadOnly Property

Indicates whether end users can use the generated control to set values.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use ReadOnly to indicate whether users can change the value displayed by the generated control. This value determines whether there is a READONLY option added to the generated TEXTAREA tag.

37.1.20.2.9 TFieldTextArea.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.20.2.10 TFieldTextArea.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.20.2.11 TFieldTextArea.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.20.2.12 TFieldTextArea.Wrap Property

Specifies how the generated control treats lines that are longer than the display width.

Pascal

```
property Wrap: TTextAreaWrap;
```

C++

```
__property TTextAreaWrap Wrap;
```

Description

Use Wrap to specify how the generated control formats lines that are longer than the number of characters specified by DisplayWidth. TWebTextAreaInput uses Wrap to add a WRAP option to the generated TEXTAREA tag.

Wrap can have any of the following values:

Value	WRAP option value	Meaning
wrOff	OFF	The control does not wrap long lines. Only newline characters that are part of the text string cause the lines in the control to wrap.
wrPhysical	PHYSICAL	The control formats its text value by wrapping long lines so that no text is hidden. The control inserts carriage returns into the text string.
wrVirtual	VIRTUAL	The control formats its text value by wrapping long lines so that no text is hidden but does not change the actual value of the text string.

See Also

DisplayWidth (see page 2144)

37.1.21 TFirstButton Class

TFirstButton generates an HTML control that moves to the first record in an XML data packet.

Class Hierarchy

Pascal

```
TFirstButton = class(TDataSetButton);
```

C++

```
class TFirstButton : public TDataSetButton;
```

File

MidItems

Description

Add TFirstButton as the child of a TDataNavigator (see page 1855) component (or a layout group that is a child of a TDataNavigator (see page 1855) component) to enable the generated HTML form to move quickly to the first record in a dataset. The resulting HTML button's action changes the associated display control so that it shows the first record as current.

Use the properties of TFirstButton to customize the appearance of the button and associate it with a Web item. The associated Web item displays the records that the first button navigates.

See Also

TXMLBroker

TDataNavigator (see page 1855)

TLayoutGroup (see page 1939)

37.1.21.1 TFirstButton Members

The following tables list the members exposed by TFirstButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.
V	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 2194)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
R	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
V	Create (see page 1915)	Creates and initializes an instance of TFirstButton (see page 1913).
V	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.
	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.21.2 TFirstButton.Create Constructor

Creates and initializes an instance of TFirstButton (see page 1913).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TFirstButton(TComponent * AOwner);
```

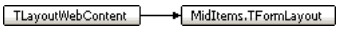
Description

Call Create to instantiate TFirstButton (see page 1913) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.22 TFormLayout Class

TFormLayout lets child Web items adhere to layout rules imposed by their parent.

Class Hierarchy



Pascal

```
TFormLayout = class(TLayoutWebContent);
```

C++

```
class TFormLayout : public TLayoutWebContent;
```

File

MidItems

Description

TFormLayout is the default implementation of the ILayoutWebContent interface for InternetExpress applications. The Web items that belong to an InetXPageProducer object pass a TFormLayout instance to their child Web items as the ParentLayout parameter of their Content method. This object provides helper functions that the child items use to fit their generated content into the parent object's HTML structure.

Use the TFormLayout properties to configure how the ILayoutWebContent methods convert HTML that child Web items generate into a final form.

See Also

TLayout

IWebContent

ILayoutWebContent

TInetXPageProducer (see page 2285)

37.1.22.1 TFormLayout Members

The following tables list the members exposed by TFormLayout.

Protected Methods

	Name	Description
	EndButtons (see page 1917)	This is EndButtons, a member of class TFormLayout.
	EndFields (see page 1917)	This is EndFields, a member of class TFormLayout.
	EndRow (see page 1918)	This is EndRow, a member of class TFormLayout.
	EndTable (see page 1918)	This is EndTable, a member of class TFormLayout.
V	ImplLayoutButton (see page 1918)	Provides the underlying implementation for the LayoutButton method.
V	ImplLayoutField (see page 1919)	Provides the underlying implementation for the LayoutField method.
V	ImplLayoutLabelAndField (see page 1919)	Provides the underlying implementation for the LayoutLabelAndField method.
V	ImplLayoutTable (see page 1920)	Provides the underlying implementation for the LayoutTable method.
	NextColumn (see page 1920)	This is NextColumn, a member of class TFormLayout.
	StartButtons (see page 1920)	This is StartButtons, a member of class TFormLayout.
	StartFields (see page 1921)	This is StartFields, a member of class TFormLayout.
	StartRow (see page 1921)	This is StartRow, a member of class TFormLayout.
	StartTable (see page 1921)	This is StartTable, a member of class TFormLayout.

Public Methods

	Name	Description
	Create (see page 1916)	Creates a new TFormLayout (see page 1915) object.
	EndLayout (see page 1917)	Returns the string of HTML to insert after laying out all child items.

Public Properties

	Name	Description
	BreakButtons (see page 1922)	Indicates whether the associated Web item uses a grid to lay out buttons that child Web items generate.
	ColumnCount (see page 1922)	Indicates the number of child elements appear in a single row when the associated Web item uses a grid to lay them out.
	TableHeader (see page 1922)	Specifies the HTML to generate at the beginning of a table that lays out child elements.

37.1.22.2 TFormLayout.Create Constructor

Creates a new TFormLayout (see page 1915) object.

Pascal

```
constructor Create(AParentLayout: TLayout);
```

C++

```
__fastcall TFormLayout(TLayout AParentLayout);
```

Description

Call Create to instantiate a TFormLayout (see page 1915) object to use when fetching the content produced by child Web items.

AParentLayout is the layout object of the parent Web item (the grandparent of the child Web items that call this object's methods). AParentLayout becomes the value of the ParentLayout property.

See Also

ParentLayout

37.1.22.3 TFormLayout Methods

The methods of the TFormLayout class are listed here.

Protected Methods

	Name	Description
	EndButtons (see page 1917)	This is EndButtons, a member of class TFormLayout.
	EndFields (see page 1917)	This is EndFields, a member of class TFormLayout.
	EndRow (see page 1918)	This is EndRow, a member of class TFormLayout.
	EndTable (see page 1918)	This is EndTable, a member of class TFormLayout.
V	ImplLayoutButton (see page 1918)	Provides the underlying implementation for the LayoutButton method.
V	ImplLayoutField (see page 1919)	Provides the underlying implementation for the LayoutField method.
V	ImplLayoutLabelAndField (see page 1919)	Provides the underlying implementation for the LayoutLabelAndField method.
V	ImplLayoutTable (see page 1920)	Provides the underlying implementation for the LayoutTable method.
	NextColumn (see page 1920)	This is NextColumn, a member of class TFormLayout.
	StartButtons (see page 1920)	This is StartButtons, a member of class TFormLayout.
	StartFields (see page 1921)	This is StartFields, a member of class TFormLayout.
	StartRow (see page 1921)	This is StartRow, a member of class TFormLayout.
	StartTable (see page 1921)	This is StartTable, a member of class TFormLayout.

Public Methods

	Name	Description
	EndLayout (see page 1917)	Returns the string of HTML to insert after laying out all child items.

37.1.22.3.1 TFormLayout.EndButtons Method

Pascal

```
function EndButtons: string;
```

C++

```
__fastcall UnicodeString EndButtons();
```

Description

This is EndButtons, a member of class TFormLayout.

37.1.22.3.2 TFormLayout.EndFields Method

Pascal

```
function EndFields: string;
```

C++

```
__fastcall UnicodeString EndFields();
```

Description

This is EndFields, a member of class TFormLayout.

37.1.22.3.3 TFormLayout.EndLayout Method

Returns the string of HTML to insert after laying out all child items.

Pascal

```
function EndLayout: string;
```

C++

```
__fastcall UnicodeString EndLayout();
```

Description

EndLayout generates the tags that must be inserted after all the content that child Web items generate. TFormLayout (see page 1915) keeps track of the ILayoutWebContent method calls that are made so that it can know what tags need to be closed.

See Also

ILayoutWebContent

37.1.22.3.4 TFormLayout.EndRow Method

Pascal

```
function EndRow: string;
```

C++

```
__fastcall UnicodeString EndRow();
```

Description

This is EndRow, a member of class TFormLayout.

37.1.22.3.5 TFormLayout.EndTable Method

Pascal

```
function EndTable: string;
```

C++

```
__fastcall UnicodeString EndTable();
```

Description

This is EndTable, a member of class TFormLayout.

37.1.22.3.6 TFormLayout.ImplLayoutButton Method

Provides the underlying implementation for the LayoutButton method.

Pascal

```
function ImplLayoutButton(const HTMLButton: string; Attributes: TLayoutAttributes): string;
override;
```

C++

```
virtual __fastcall UnicodeString ImplLayoutButton(const UnicodeString HTMLButton,
TLayoutAttributes Attributes);
```

Description

The LayoutButton method calls ImplLayoutButton to convert the HTML that declares a button (typically an <INPUT> tag) into the form that the associated Web item requires.

HTMLButton is the HTML that declares a button. This is produced by the child Web item.

Attributes describes any style attributes or layout considerations that should be applied to the final HTML for the button.

LayoutButton returns the HTML for the button in a form that the associated Web item can use.

In TLayoutWebContent, ImplLayoutButton is abstract or, in C++ terminology, pure virtual. Descendant classes override this method to provide an implementation for the LayoutButton method.

See Also

LayoutButton

37.1.22.3.7 TFormLayout.ImplLayoutField Method

Provides the underlying implementation for the LayoutField method.

Pascal

```
function ImplLayoutField(const HTMLField: string; Attributes: TLayoutAttributes): string;
override;
```

C++

```
virtual __fastcall UnicodeString ImplLayoutField(const UnicodeString HTMLField,
TLayoutAttributes Attributes);
```

Description

The LayoutField method calls ImplLayoutField to convert the HTML that declares a field (an input control that represents a data field or a named value) into the form that the associated Web item requires.

HTMLField is the HTML that declares the field. This is produced by the child Web item.

Attributes describes any style attributes or layout considerations that should be applied to the final HTML for the field.

LayoutField returns the HTML for the field in a form that the associated Web item can use.

In TLayoutWebContent, ImplLayoutField is abstract or, in C++ terminology, pure virtual. Descendant classes override this method to provide an implementation for the LayoutField method.

See Also

LayoutField

37.1.22.3.8 TFormLayout.ImplLayoutLabelAndField Method

Provides the underlying implementation for the LayoutLabelAndField method.

Pascal

```
function ImplLayoutLabelAndField(const HTMLLabel: string; const HTMLField: string; Attributes:
TLayoutAttributes): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplLayoutLabelAndField(const UnicodeString HTMLLabel, const
UnicodeString HTMLField, TLayoutAttributes Attributes);
```

Description

The LayoutLabelAndField method calls ImplLayoutLabelAndField to convert the HTML that declares a labeled field (an input control that represents a data field or a named value along with its label) into the form that the associated Web item requires.

HTMLLabel is the HTML that declares the label. This is produced by the child Web item.

HTMLField is the HTML that declares the field. This is produced by the child Web item.

Attributes describes any style attributes or layout considerations that should be applied to the final HTML for the field and its label.

LayoutLabelAndField returns the HTML for the field and label in a form that the associated Web item can use.

In TLayoutWebContent, ImplLayoutLabelAndField is abstract or, in C++ terminology, pure virtual. Descendant classes override this method to provide an implementation for the LayoutLabelAndField method.

See Also

LayoutLabelAndField

37.1.22.3.9 TFormLayout.ImplLayoutTable Method

Provides the underlying implementation for the LayoutTable method.

Pascal

```
function ImplLayoutTable(const HTMLTable: string; Attributes: TLayoutAttributes): string;
override;
```

C++

```
virtual __fastcall UnicodeString ImplLayoutTable(const UnicodeString HTMLTable,
TLayoutAttributes Attributes);
```

Description

The LayoutTable method calls ImplLayoutTable to convert the HTML that declares a table (typically a structure used to layout the content produced by the child Web item's children) into the form that the associated Web item requires.

HTMLTable is the HTML that declares the table. This is produced by the child Web item.

Attributes describes any style attributes or layout considerations that should be applied to the final HTML for the table.

LayoutTable returns the HTML for the table in a form that the associated Web item can use.

In TLayoutWebContent, ImplLayoutTable is abstract or, in C++ terminology, pure virtual. Descendant classes override this method to provide an implementation for the LayoutTable method.

See Also

LayoutTable

37.1.22.3.10 TFormLayout.NextColumn Method

Pascal

```
function NextColumn(SubItemCount: Integer): string;
```

C++

```
__fastcall UnicodeString NextColumn(int SubItemCount);
```

Description

This is NextColumn, a member of class TFormLayout.

37.1.22.3.11 TFormLayout.StartButtons Method

Pascal

```
function StartButtons: string;
```

C++

```
__fastcall UnicodeString StartButtons();
```

Description

This is StartButtons, a member of class TFormLayout.

37.1.22.3.12 TFormLayout.StartFields Method

Pascal

```
function StartFields(ColCount: Integer): string;
```

C++

```
__fastcall UnicodeString StartFields(int ColCount);
```

Description

This is StartFields, a member of class TFormLayout.

37.1.22.3.13 TFormLayout.StartRow Method

Pascal

```
function StartRow: string;
```

C++

```
__fastcall UnicodeString StartRow();
```

Description

This is StartRow, a member of class TFormLayout.

37.1.22.3.14 TFormLayout.StartTable Method

Pascal

```
function StartTable: string;
```

C++

```
__fastcall UnicodeString StartTable();
```

Description

This is StartTable, a member of class TFormLayout.

37.1.22.4 TFormLayout Properties

The properties of the TFormLayout class are listed here.

Public Properties

	Name	Description
	BreakButtons (see page 1922)	Indicates whether the associated Web item uses a grid to lay out buttons that child Web items generate.
	ColumnCount (see page 1922)	Indicates the number of child elements appear in a single row when the associated Web item uses a grid to lay them out.
	TableHeader (see page 1922)	Specifies the HTML to generate at the beginning of a table that lays out child elements.

37.1.22.4.1 TFormLayout.BreakButtons Property

Indicates whether the associated Web item uses a grid to lay out buttons that child Web items generate.

Pascal

```
property BreakButtons: Boolean;
```

C++

```
__property Boolean BreakButtons;
```

Description

Set BreakButtons to true if the Web item that supplies this object to its child Web items uses an HTML table to position HTML buttons that those child Web items generate. Set BreakButtons to false if the Web item does not use an HTML table to position child buttons.

BreakButtons affects the whether the LayoutButton method on this object's ILayoutWebContent interface adds <TD> tags to the HTML that child Web items generate for HTML buttons.

See Also

LayoutButton

37.1.22.4.2 TFormLayout.ColumnCount Property

Indicates the number of child elements appear in a single row when the associated Web item uses a grid to lay them out.

Pascal

```
property ColumnCount: Integer;
```

C++

```
__property int ColumnCount;
```

Description

Set ColumnCount to the number of child elements that appear in a single row when laying out child controls by rows.

ColumnCount affects where the LayoutField or LayoutButton method on this object's ILayoutWebContent interface adds <TR> and </TR> tags to the HTML that child Web items generate.

Note: ColumnCount only affects the LayoutButton method if the value of the BreakButtons (see page 1922) property is true.

See Also

LayoutButton

LayoutField

BreakButtons (see page 1922)

37.1.22.4.3 TFormLayout.TableHeader Property

Specifies the HTML to generate at the beginning of a table that lays out child elements.

Pascal

```
property TableHeader: string;
```

C++

```
__property UnicodeString TableHeader;
```

Description

Set TableHeader to the HTML tag that appears at the beginning of the HTML table used to control the layout of child elements. TableHeader is a string that includes the <table> tag, along with any attributes (such as alignment) that affect the layout of child items.

See Also

LayoutButton

LayoutField

BreakButtons (see page 1922)

37.1.23 TGridAttributes Class

TGridAttributes describes the options the TDataGrid (see page 1850) component adds to an HTML <TABLE> tag.

Class Hierarchy

Pascal

```
TGridAttributes = class(TPersistent);
```

C++

```
class TGridAttributes : public TPersistent;
```

File

MidItems

Description

TGridAttributes implements the TableAttributes property of a TDataGrid (see page 1850) or TAdapterGrid component. It describes display attributes that apply to an HTML table generated by TDataGrid (see page 1850) or TAdapterGrid. The properties of TGridAttributes specify the options added to the generated HTML <TABLE> tag, as well as any styles that should be applied.

See Also

TCustomDataGrid (see page 1811)

TCustomAdapterGrid

37.1.23.1 TGridAttributes Members

The following tables list the members exposed by TGridAttributes.

Protected Methods

	Name	Description
	AssignTo (see page 1924)	Copies the properties of an object to a destination object.

Public Methods

	Name	Description
	Create (see page 1924)	Creates an instance of TGridAttributes (see page 1923).

Public Properties

	Name	Description
	Parent (see page 1928)	Indicates the TDataGrid (see page 1850) object that uses this TGridAttributes (see page 1923) to describe its attributes.

Published Properties

	Name	Description
	Align (see page 1926)	Specifies the alignment of the table within the HTML document.
	BgColor (see page 1926)	Specifies the background color of the HTML table.
	Border (see page 1926)	Specifies the thickness of the border drawn around the HTML table.
	CellPadding (see page 1927)	Specifies the amount of space to leave around the contents of each cell in the HTML table.
	CellSpacing (see page 1927)	Specifies the amount of space to leave between cells in the HTML table.
	Custom (see page 1928)	Represents a tag option and its value in an HTML tag.
	Style (see page 1928)	Adds style attributes to the generated table.
	StyleRule (see page 1929)	Applies a style in the page producer's style sheet to the generated table.

37.1.23.2 TGridAttributes.Create Constructor

Creates an instance of TGridAttributes (see page 1923).

Pascal

```
constructor Create(AParent: TComponent);
```

C++

```
__fastcall TGridAttributes(TComponent * AParent);
```

Description

Most applications have no need to instantiate TGridAttributes (see page 1923). TGridAttributes (see page 1923) is instantiated by the TDataGrid (see page 1850) component that uses it.

The AParent parameter is the TDataGrid (see page 1850) or TAdapterGrid component that uses the TGridAttributes (see page 1923) instance. This parameter becomes the value of the Parent (see page 1928) property.

See Also

Parent (see page 1928)

TDataGrid (see page 1850)

TAdapterGrid

37.1.23.3 TGridAttributes Methods

The methods of the TGridAttributes class are listed here.

Protected Methods

	Name	Description
	AssignTo (see page 1924)	Copies the properties of an object to a destination object.

37.1.23.3.1 TGridAttributes.AssignTo Method

Copies the properties of an object to a destination object.

Pascal

```
procedure AssignTo(Dest: TPersistent); override;
```

C++

```
virtual __fastcall AssignTo(TPersistent * Dest);
```

Description

Override the AssignTo method to extend the functionality of the Assign method of destination objects so that they handle newly created object classes. When defining a new object class, override the Assign method for every existing object class that should be able to copy its properties to the new class. Override the AssignTo method for every existing class to which the new class can copy.

The Assign method of TPersistent calls AssignTo if the descendant object does not succeed in copying the properties of a source object. The AssignTo method defined by TPersistent raises an EConvertError exception.

For example, given the following code in which A and B are instance variables:

```
A.Assign(B); {Delphi}
A->Assign(B); // C++
```

if A knows how to handle B, then it does so and returns. If A doesn't know how to handle B's type, execution will trickle to the TPersistent version of Assign, which calls:

```
B.AssignTo(A); {Delphi}
B->AssignTo(A); // C++
```

If B knows how to copy to A, the assignment succeeds. Otherwise, TPersistent raises an exception.

See Also

Assign

EConvertError

37.1.23.4 TGridAttributes Properties

The properties of the TGridAttributes class are listed here.

Public Properties

	Name	Description
R	Parent (see page 1928)	Indicates the TDataGrid (see page 1850) object that uses this TGridAttributes (see page 1923) to describe its attributes.

Published Properties

	Name	Description
	Align (see page 1926)	Specifies the alignment of the table within the HTML document.
	BgColor (see page 1926)	Specifies the background color of the HTML table.
	Border (see page 1926)	Specifies the thickness of the border drawn around the HTML table.
	CellPadding (see page 1927)	Specifies the amount of space to leave around the contents of each cell in the HTML table.
	CellSpacing (see page 1927)	Specifies the amount of space to leave between cells in the HTML table.
	Custom (see page 1928)	Represents a tag option and its value in an HTML tag.
	Style (see page 1928)	Adds style attributes to the generated table.
	StyleRule (see page 1929)	Applies a style in the page producer's style sheet to the generated table.

37.1.23.4.1 TGridAttributes.Align Property

Specifies the alignment of the table within the HTML document.

Pascal

```
property Align: THTMLAlign;
```

C++

```
__property THTMLAlign Align;
```

Description

Set Align to indicate the desired alignment of the table within the HTML document, or within the cell of a surrounding HTML table in a layout group. Align indicates the desired value of the Align option of the HTML table tag.

The default value of Align is haDefault, which means the web browser that reads the HTML document positions the HTML table where it wants.

See Also

CellPadding (see page 1927)

Border (see page 1926)

37.1.23.4.2 TGridAttributes.BgColor Property

Specifies the background color of the HTML table.

Pascal

```
property BgColor: THTMLBgColor;
```

C++

```
__property THTMLBgColor BgColor;
```

Description

Use BgColor to indicate the desired background color of the HTML table. The value of BgColor is the name of color. This can be one of the following strings:

Aqua	Black	Blue	Fuchsia
Gray	Green	Lime	Maroon
Navy	Olive	Purple	Red
Silver	Teal	White	Yellow

Note: The row or heading attributes can override this background color for a portion of the HTML table.

See Also

BgColor (see page 1932)

37.1.23.4.3 TGridAttributes.Border Property

Specifies the thickness of the border drawn around the HTML table.

Pascal

```
property Border: Integer;
```

C++

```
__property int Border;
```

Description

Set Border to specify whether a border should be drawn around the table, and if so, how thick it should be. The thickness specified by any particular value of border depends on the web browser that views the HTML table. For any browser, however, larger numbers indicate thicker borders. A value of -1 indicates that no border should be drawn around the table.

See Also

CellSpacing (see page 1927)

37.1.23.4.4 TGridAttributes.CellPadding Property

Specifies the amount of space to leave around the contents of each cell in the HTML table.

Pascal

```
property CellPadding: Integer;
```

C++

```
__property int CellPadding;
```

Description

Set CellPadding to specify how many pixels of whitespace should appear between a cell's contents and the edges of the cell. A value of -1 indicates that the web browser should decide how to pad the cells of the HTML table.

See Also

CellSpacing (see page 1927)

37.1.23.4.5 TGridAttributes.CellSpacing Property

Specifies the amount of space to leave between cells in the HTML table.

Pascal

```
property CellSpacing: Integer;
```

C++

```
__property int CellSpacing;
```

Description

Set CellSpacing to specify how many pixels of whitespace should appear between the cells in the table. A value of -1 indicates that the web browser should decide how to separate the cells of the HTML table.

Note: CellSpacing refers to the whitespace between cells in an HTML table, not the thickness of the lines drawn to separate rows and columns.

See Also

Border (see page 1926)

CellPadding (see page 1927)

37.1.23.4.6 TGridAttributes.Custom Property

Represents a tag option and its value in an HTML tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

Use Custom to represent one or more options to be added to the HTML <TABLE> tag. Each option and its value should have the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to hide the side borders and add row dividers, you can set Custom to

```
FRAME=HSIDES RULES=ROWS
```

See Also

Style (see page 1928)

StyleRule (see page 1929)

37.1.23.4.7 TGridAttributes.Parent Property

Indicates the TDataGrid (see page 1850) object that uses this TGridAttributes (see page 1923) to describe its attributes.

Pascal

```
property Parent: TComponent;
```

C++

```
__property TComponent * Parent;
```

Description

Read Parent to access the TDataGrid (see page 1850) that uses this TGridAttributes (see page 1923) object. The value of Parent is assigned by the constructor.

See Also

TDataGrid (see page 1850)

Create (see page 1924)

TAdapterGrid

37.1.23.4.8 TGridAttributes.Style Property

Adds style attributes to the generated table.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the entire table. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the <TABLE> tag as the value of a STYLE option.

See Also

Custom (see page 1928)

StyleRule (see page 1929)

37.1.23.4.9 TGridAttributes.StyleRule Property

Applies a style in the page producer's style sheet to the generated table.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated table. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style (see page 1928) property instead.

See Also

Style (see page 1928)

Styles (see page 2272)

StylesFile (see page 2273)

Styles

StylesFile

37.1.24 TGridRowAttributes Class

TGridRowAttributes describes the options the TDataGrid (see page 1850) component adds to its HTML row tags (<TR>).

Class Hierarchy

Pascal

```
TGridRowAttributes = class(TPersistent);
```

C++

```
class TGridRowAttributes : public TPersistent;
```

File

MidItems

Description

TGridRowAttributes implements the RowAttributes and HeadingAttributes properties of a TDataGrid (see page 1850) or TAdapterGrid component. It describes display attributes that apply to the generated HTML table's rows. The properties of TGridRowAttributes specify the options added to the generated HTML <TR> tag, as well as any styles that should be applied to the header or body of the table.

See Also

TCustomDataGrid (see page 1811)

TCustomAdapterGrid

37.1.24.1 TGridRowAttributes Members

The following tables list the members exposed by TGridRowAttributes.

Protected Methods

	Name	Description
	AssignTo (see page 1931)	Copies the properties of an object to a destination object.

Public Methods

	Name	Description
	Create (see page 1930)	Creates an instance of TGridRowAttributes (see page 1929).

Public Properties

	Name	Description
	Parent (see page 1933)	Indicates the TDataGrid (see page 1850) or TAdapterGrid object that uses this TGridRowAttributes (see page 1929) to describe its header or row attributes.

Published Properties

	Name	Description
	Align (see page 1932)	Specifies the alignment of the table within the HTML document.
	BgColor (see page 1932)	Specifies the background color of the rows or header of an HTML table.
	Custom (see page 1933)	Represents a tag option and its value in an HTML tag.
	Style (see page 1934)	Adds style attributes to the generated table body or header.
	StyleRule (see page 1934)	Applies a style in the page producer's style sheet to the generated table body or header.
	VAlign (see page 1934)	Specifies the vertical alignment within the cells of the HTML table body or header.

37.1.24.2 TGridRowAttributes.Create Constructor

Creates an instance of TGridRowAttributes (see page 1929).

Pascal

```
constructor Create(AParent: TComponent);
```

C++

```
__fastcall TGridRowAttributes(TComponent * AParent);
```

Description

Most applications have no need to instantiate TGridRowAttributes (see page 1929). TGridRowAttributes (see page 1929) is

instantiated by the TDataGrid (see page 1850) or TAdapterGrid component that uses it.

The AParent parameter is the TDataGrid (see page 1850) or TAdapterGrid component that uses the TGridRowAttributes (see page 1929) instance. This parameter becomes the value of the Parent (see page 1933) property.

See Also

Parent (see page 1933)

TDataGrid (see page 1850)

TAdapterGrid

37.1.24.3 TGridRowAttributes Methods

The methods of the TGridRowAttributes class are listed here.

Protected Methods

	Name	Description
	AssignTo (see page 1931)	Copies the properties of an object to a destination object.

37.1.24.3.1 TGridRowAttributes.AssignTo Method

Copies the properties of an object to a destination object.

Pascal

```
procedure AssignTo(Dest: TPersistent); override;
```

C++

```
virtual __fastcall AssignTo(TPersistent * Dest);
```

Description

Override the AssignTo method to extend the functionality of the Assign method of destination objects so that they handle newly created object classes. When defining a new object class, override the Assign method for every existing object class that should be able to copy its properties to the new class. Override the AssignTo method for every existing class to which the new class can copy.

The Assign method of TPersistent calls AssignTo if the descendant object does not succeed in copying the properties of a source object. The AssignTo method defined by TPersistent raises an EConvertError exception.

For example, given the following code in which A and B are instance variables:

```
A.Assign(B); {Delphi}
A->Assign(B); // C++
```

if A knows how to handle B, then it does so and returns. If A doesn't know how to handle B's type, execution will trickle to the TPersistent version of Assign, which calls:

```
B.AssignTo(A); {Delphi}
B->AssignTo(A); // C++
```

If B knows how to copy to A, the assignment succeeds. Otherwise, TPersistent raises an exception.

See Also

Assign

EConvertError

37.1.24.4 TGridRowAttributes Properties

The properties of the TGridRowAttributes class are listed here.

Public Properties

	Name	Description
	Parent (see page 1933)	Indicates the TDataGrid (see page 1850) or TAdapterGrid object that uses this TGridRowAttributes (see page 1929) to describe its header or row attributes.

Published Properties

	Name	Description
	Align (see page 1932)	Specifies the alignment of the table within the HTML document.
	BgColor (see page 1932)	Specifies the background color of the rows or header of an HTML table.
	Custom (see page 1933)	Represents a tag option and its value in an HTML tag.
	Style (see page 1934)	Adds style attributes to the generated table body or header.
	StyleRule (see page 1934)	Applies a style in the page producer's style sheet to the generated table body or header.
	VAlign (see page 1934)	Specifies the vertical alignment within the cells of the HTML table body or header.

37.1.24.4.1 TGridRowAttributes.Align Property

Specifies the alignment of the table within the HTML document.

Pascal

```
property Align: THTMLAlign;
```

C++

```
__property THTMLAlign Align;
```

Description

Set Align to indicate the desired alignment of the contents of each cell in the table row or header. Align indicates the desired value of the Align option of the <TR> tag.

The default value of Align is haDefault, which means the web browser that reads the HTML document positions the cell contents wherever it wants horizontally.

See Also

VAlign (see page 1934)

37.1.24.4.2 TGridRowAttributes.BgColor Property

Specifies the background color of the rows or header of an HTML table.

Pascal

```
property BgColor: THTMLBgColor;
```

C++

```
__property THTMLBgColor BgColor;
```

Description

Use BgColor to indicate the desired background color of the rows or heading of an HTML table. The value of BgColor is the name of color. This can be one of the following strings:

Aqua	Black	Blue	Fuchsia
Gray	Green	Lime	Maroon
Navy	Olive	Purple	Red
Silver	Teal	White	Yellow

Note: This property overrides the background color option of the HTML table itself.

See Also

BgColor (see page 1926)

37.1.24.4.3 TGridRowAttributes.Custom Property

Represents a tag option and its value in an HTML tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

Use Custom to represent one or more options to be added to the HTML <TR> tag. Each option and its value should have the form

```
OPTIONNAME=Value
```

To specify multiple options, separate the Name/value pairs with spaces.

See Also

Style (see page 1934)

StyleRule (see page 1934)

37.1.24.4.4 TGridRowAttributes.Parent Property

Indicates the TDataGrid (see page 1850) or TAdapterGrid object that uses this TGridRowAttributes (see page 1929) to describe its header or row attributes.

Pascal

```
property Parent: TComponent;
```

C++

```
__property TComponent * Parent;
```

Description

Read Parent to access the TDataGrid (see page 1850) or TAdapterGrid that uses this TGridRowAttributes (see page 1929) object. The value of Parent is assigned by the constructor.

See Also

TDataGrid (see page 1850)

TAdapterGrid

Create (see page 1930)

37.1.24.4.5 TGridRowAttributes.Style Property

Adds style attributes to the generated table body or header.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the rows or header of the HTML table. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the <TR> tag as the value of a STYLE option.

See Also

Custom (see page 1933)

StyleRule (see page 1934)

37.1.24.4.6 TGridRowAttributes.StyleRule Property

Applies a style in the page producer's style sheet to the generated table body or header.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated table body or header. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style (see page 1934) property instead.

See Also

Style (see page 1934)

Styles (see page 2272)

StylesFile (see page 2273)

Styles

StylesFile

37.1.24.4.7 TGridRowAttributes.VAlign Property

Specifies the vertical alignment within the cells of the HTML table body or header.

Pascal

```
property VAlign: THTMLVAlign;
```

C++

```
__property THTMLVAlign VAlign;
```

Description

Set VAlign to indicate the desired vertical alignment of the contents of each cell. VAlign indicates the desired value of the VAlign option of the <TR> tag for the table rows or header.

The default value of VAlign is haVDefault, which means the web browser that reads the HTML document positions the contents of cells in the table element wherever it wishes vertically.

See Also

Align (see page 1932)

37.1.25 TInsertButton Class

TInsertButton generates an HTML control that inserts a new record in an XML data packet.

Class Hierarchy

Pascal

```
TInsertButton = class(TDataSetButton);
```

C++

```
class TInsertButton : public TDataSetButton;
```

File

MidItems

Description

Add TInsertButton as the child of a TDataNavigator (see page 1855) component (or a layout group that is a child of a TDataNavigator (see page 1855) component) to enable the generated HTML form to add records to an XML data packet. The resulting HTML button's action adds a new blank record to the emerging XML delta packet and causes the associated display control to display that new record.

Use the properties of TInsertButton to customize the appearance of the button and associate it with a Web item. The associated Web item generates the HTML control that displays the newly inserted record.

See Also

TXMLBroker

TDataNavigator (see page 1855)

TLayoutGroup (see page 1939)

TApplyUpdatesButton (see page 1798)

37.1.25.1 TInsertButton Members

The following tables list the members exposed by TInsertButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.
V	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 2194)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
R	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
V	Create (see page 1937)	Creates and initializes an instance of TInsertButton (see page 1935).
V	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.

	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.25.2 TInsertButton.Create Constructor

Creates and initializes an instance of TInsertButton (see page 1935).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TInsertButton(TComponent * AOwner);
```

Description

Call Create to instantiate TInsertButton (see page 1935) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.26 TLastButton Class

TLastButton generates an HTML control that moves to the last record in an XML data packet.

Class Hierarchy

Pascal

```
TLastButton = class(TDataSetButton);
```

C++

```
class TLastButton : public TDataSetButton;
```

File

MidItems

Description

Add TLastButton as the child of a TDataNavigator (see page 1855) component (or a layout group that is a child of a TDataNavigator (see page 1855) component) to enable the generated HTML form to move quickly to the last record in a dataset. The resulting HTML button's action changes the associated display control so that it shows the last record as current.

Use the properties of TLastButton to customize the appearance of the button and associate it with a Web item. The associated Web item displays the records that the last button navigates.

See Also

TXMLBroker

TDataNavigator (see page 1855)

TLayoutGroup (see page 1939)

37.1.26.1 TLastButton Members

The following tables list the members exposed by TLastButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.
V	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 2194)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
R	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
V	Create (see page 1939)	Creates and initializes an instance of TLastButton (see page 1937).
V	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.
	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.26.2 TLastButton.Create Constructor

Creates and initializes an instance of TLastButton (see page 1937).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TLastButton(TComponent * AOwner);
```

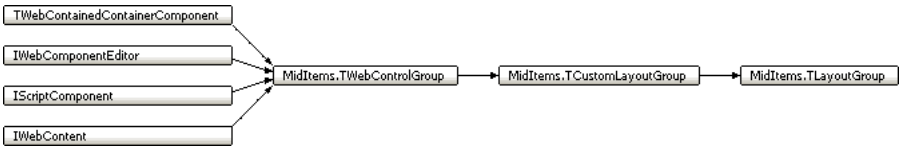
Description

Call Create to instantiate TLastButton (see page 1937) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.27 TLayoutGroup Class

TLayoutGroup generates the HTML to arrange content produced by other Web items.

Class Hierarchy



Pascal

```
TLayoutGroup = class(TCustomLayoutGroup);
```

C++

```
class TLayoutGroup : public TCustomLayoutGroup;
```

File

MidItems

Description

Use TLayoutGroup to adjust the layout of items in an HTML document. There are few restrictions on what Web items can use a layout group because it has no special content that it displays. It merely imposes a layout on content generated by other Web items.

TLayoutGroup maintains a list of Web items that display information in the final HTML document, but imposes no restrictions of its own on the possible types of Web items. Instead, it restricts those Web items to the classes that its parent allows. In order to

let the parent component use the HTML that TLayoutGroup generates, TLayoutGroup inherits the IWebContent interface from TWebControlGroup (see page 2064).

See Also

IWebContent

Using the Web Page Editor

37.1.27.1 TLayoutGroup Members

The following tables list the members exposed by TLayoutGroup.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
V	GetLayoutAttributes (see page 1832)	Returns an object that specifies how the generated group of controls should look.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
V	ImplAddElements (see page 2069)	Provides the underlying implementation for the AddElements (see page 2066) method.
V	ImplCanAddClass (see page 1833)	Provides the underlying implementation for the CanAddClass method.
V	ImplCanAddClass (see page 2070)	Provides the underlying implementation for the CanAddClass (see page 2067) method.
V	ImplContent (see page 1833)	Provides the underlying implementation for the Content method.
V	ImplContent (see page 2070)	Provides the underlying implementation for the Content (see page 2068) method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.

Public Methods

	Name	Description
V	Create (see page 1832)	Creates and initializes an instance of TCustomLayoutGroup (see page 1830).
V	Create (see page 2065)	Creates and initializes an instance of TWebControlGroup (see page 2064).
V	Destroy (see page 1832)	Frees an instance of TCustomLayoutGroup (see page 1830).
V	Destroy (see page 2066)	Frees an instance of TWebControlGroup (see page 2064).

Public Properties

	Name	Description
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.

Published Properties

	Name	Description
	Custom (see page 1941)	Represents a tag option and its value in the generated TABLE tag.

	DisplayColumns (see page 1941)	Specifies the number of columns the layout group uses to lay out its child controls.
	Style (see page 1942)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1942)	Applies a style in the page producer's style sheet to the group of buttons.

37.1.27.2 TLayoutGroup Properties

The properties of the TLayoutGroup class are listed here.

Published Properties

	Name	Description
	Custom (see page 1941)	Represents a tag option and its value in the generated TABLE tag.
	DisplayColumns (see page 1941)	Specifies the number of columns the layout group uses to lay out its child controls.
	Style (see page 1942)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1942)	Applies a style in the page producer's style sheet to the group of buttons.

37.1.27.2.1 TLayoutGroup.Custom Property

Represents a tag option and its value in the generated TABLE tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TCustomLayoutGroup generates an HTML table to lay out the controls generated by the Web items in its WebFieldControls property. Use Custom to add options that qualify the TABLE tag. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to add a border around the group of controls and color the background blue, you can set Custom to

```
BORDER=2 BGCOLOR=BLUE
```

See Also

Style (see page 1835)

StyleRule (see page 1835)

WebFieldControls (see page 2072)

DisplayColumns (see page 1834)

37.1.27.2.2 TLayoutGroup.DisplayColumns Property

Specifies the number of columns the layout group uses to lay out its child controls.

Pascal

```
property DisplayColumns: Integer;
```

C++

```
__property int DisplayColumns;
```

Description

TCustomLayoutGroup generates an HTML table to lay out the controls generated by the Web items in its WebFieldControls property. It adds the HTML generated by each Web item listed in WebFieldControls to a cell of the table, filling the cells out row by row until it displays all the Web items. DisplayColumns determines the number of cells in each row.

See Also

WebFieldControls (see page 2072)

37.1.27.2.3 TLayoutGroup.Style Property

Adds style attributes to the generated HTML table.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

TCustomLayoutGroup generates an HTML table to lay out the controls generated by the Web items in WebFieldControls. Use Style to add a STYLE option that qualifies the TABLE tag. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated TABLE tag as the value of a STYLE option.

See Also

Custom (see page 1834)

StyleRule (see page 1835)

WebFieldControls (see page 2072)

37.1.27.2.4 TLayoutGroup.StyleRule Property

Applies a style in the page producer's style sheet to the group of buttons.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

TCustomLayoutGroup generates an HTML table to lay out the controls generated by its Web items. Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated table. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the Style property.

See Also

Style (see page 1835)

Styles

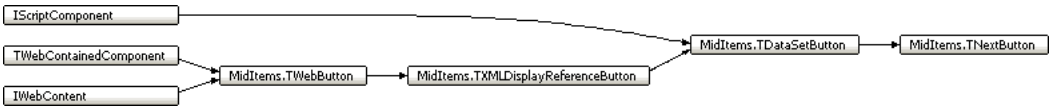
StylesFile

WebFieldControls (see page 2072)

37.1.28 TNextButton Class

TNextButton generates an HTML control that moves to the next record in an XML data packet.

Class Hierarchy



Pascal

```
TNextButton = class(TDataSetButton);
```

C++

```
class TNextButton : public TDataSetButton;
```

File

MidItems

Description

Add TNextButton as the child of a TDataNavigator (see page 1855) component (or a layout group that is a child of a TDataNavigator (see page 1855) component) to enable an associated control to move to the next record in a dataset. The resulting HTML button's action changes the current record displayed by the associated control.

Use the properties of TNextButton to customize the appearance of the button and associate it with a Web item. The associated Web item displays the records that the next button navigates.

See Also

TXMLBroker

TDataNavigator (see page 1855)

TLayoutGroup (see page 1939)

37.1.28.1 TNextButton Members

The following tables list the members exposed by TNextButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.

	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.
V	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 2194)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
R	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
V	Create (see page 1944)	Creates and initializes an instance of TNextButton (see page 1943).
V	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.
	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.28.2 TNextButton.Create Constructor

Creates and initializes an instance of TNextButton (see page 1943).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TNextButton(TComponent * AOwner);
```

Description

Call Create to instantiate TNextButton (see page 1943) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.29 TNextPageButton Class

TNextPageButton generates an HTML control that moves to the next set of records in an XML data packet.

Class Hierarchy

Pascal

```
TNextPageButton = class(TDataSetButton);
```

C++

```
class TNextPageButton : public TDataSetButton;
```

File

MidItems

Description

Add TNextPageButton as the child of a TDataNavigator (see page 1855) component (or a layout group that is a child of a TDataNavigator (see page 1855) component) to enable the generated HTML form to display the next group of records in a dataset. If the button is associated with a grid (TDataGrid (see page 1850)), the grid scrolls by the number of rows in the grid. If the button is associated with a collection of fields (TFieldGroup (see page 1869)), the current record changes to the next record in the data packet.

Use the properties of TNextPageButton to customize the appearance of the button and associate it with a display component.

See Also

TXMLBroker

TDataNavigator (see page 1855)

TLayoutGroup (see page 1939)

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

37.1.29.1 TNextPageButton Members

The following tables list the members exposed by TNextPageButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.
V	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 2194)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
R	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
V	Create (see page 1947)	Creates and initializes an instance of TNextPageButton (see page 1945).
V	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.
	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.29.2 TNextPageButton.Create Constructor

Creates and initializes an instance of TNextPageButton (see page 1945).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TNextPageButton(TComponent * AOwner);
```

Description

Call Create to instantiate TNextPageButton (see page 1945) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.30 TPostButton Class

TPostButton generates an HTML control that posts changes to an XML delta packet.

Class Hierarchy

Pascal

```
TPostButton = class(TDataSetButton);
```

C++

```
class TPostButton : public TDataSetButton;
```

File

MidItems

Description

Add TPostButton as the child of a TDataNavigator (see page 1855) component (or a layout group that is a child of a TDataNavigator (see page 1855) component) to enable the generated HTML form to commit the edits on the current record of an associated display control. The resulting HTML button's action posts all edits to the current record to the emerging delta packet, which can then be submitted using an apply updates button.

Use the properties of TPostButton to customize the appearance of the button and associate it with a Web item that generates the display control.

See Also

TXMLBroker

TDataNavigator (see page 1855)

TLayoutGroup (see page 1939)

TApplyUpdatesButton (see page 1798)

37.1.30.1 TPostButton Members

The following tables list the members exposed by TPostButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.
V	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 2194)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
R	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
V	Create (see page 1949)	Creates and initializes an instance of TPostButton (see page 1947).
V	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.

	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.30.2 TPostButton.Create Constructor

Creates and initializes an instance of TPostButton (see page 1947).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TPostButton(TComponent * AOwner);
```

Description

Call Create to instantiate TPostButton (see page 1947) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.31 TPriorButton Class

TPriorButton generates an HTML control that moves to the previous record in an XML data packet.

Class Hierarchy

Pascal

```
TPriorButton = class(TDataSetButton);
```

C++

```
class TPriorButton : public TDataSetButton;
```

File

MidItems

Description

Add TPriorButton as the child of a TDataNavigator (see page 1855) component (or a layout group that is a child of a TDataNavigator (see page 1855) component) to enable an associated control to move to the previous record in a dataset. The resulting HTML button's action changes the current record displayed by the associated control.

Use the properties of TPriorButton to customize the appearance of the button and associate it with a Web item. The associated Web item displays the records that the prior button navigates.

See Also

TXMLBroker

TDataNavigator (see page 1855)

TLayoutGroup (see page 1939)

37.1.31.1 TPriorButton Members

The following tables list the members exposed by TPriorButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.
V	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 2194)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
R	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
V	Create (see page 1951)	Creates and initializes an instance of TPriorButton (see page 1949).
V	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.
	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.31.2 TPriorButton.Create Constructor

Creates and initializes an instance of TPriorButton (see page 1949).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TPriorButton(TComponent * AOwner);
```

Description

Call Create to instantiate TPriorButton (see page 1949) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.32 TPriorPageButton Class

TPriorPageButton generates an HTML control that moves to the previous set of records in an XML data packet.

Class Hierarchy

Pascal

```
TPriorPageButton = class(TDataSetButton);
```

C++

```
class TPriorPageButton : public TDataSetButton;
```

File

MidItems

Description

Add TPriorPageButton as the child of a TDataNavigator (see page 1855) component (or a layout group that is a child of a TDataNavigator (see page 1855) component) to enable the generated HTML form to display the previous group of records in a dataset. If the button is associated with a grid (TDataGrid (see page 1850)), the grid scrolls by the number of rows in the grid. If the button is associated with a collection of fields (TFieldGroup (see page 1869)), the current record changes to the previous record in the data packet.

Use the properties of TNextPageButton (see page 1945) to customize the appearance of the button and associate it with a

display component.

See Also

TXMLBroker

TDataNavigator (see page 1855)

TLayoutGroup (see page 1939)

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

37.1.32.1 TPriorPageButton Members

The following tables list the members exposed by TPriorPageButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.
V	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 2194)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
R	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
V	Create (see page 1953)	Creates and initializes an instance of TPriorPageButton (see page 1951).
V	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.
	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.32.2 TPriorPageButton.Create Constructor

Creates and initializes an instance of TPriorPageButton (see page 1951).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TPriorPageButton(TComponent * AOwner);
```

Description

Call Create to instantiate TPriorPageButton (see page 1951) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.33 TQueryButton Class

TQueryButton is the base class for Web items that create HTML buttons that manage application-defined values.

Class Hierarchy

Pascal

```
TQueryButton = class(TWebButton);
```

C++

```
class TQueryButton : public TWebButton;
```

File

MidItems

Description

Use TQueryButton as a base class in an InternetExpress application for components that generate the HTML statements defining a specialized button control for managing the application-defined input controls. TQueryButton objects are used by TQueryForm (see page 1971), which organizes the generated buttons and the input controls they manipulate.

TQueryButton defines properties and methods for

Generating the HTML code to define the button control.

Specifying tag attributes of the HTML button control, including those defined by a style sheet.

Associating the button with a Query form whose controls it influences.

Fitting the component into a hierarchy of Web items, each of which generates part of a larger HTML document.

In order to allow other Web items to use the HTML generated by the button control, TQueryButton inherits support for the IWebContent interface from TWebButton (see page 2058).

Do not create instances of TQueryButton. TQueryButton relies on anabstract or, in C++ terminology, pure virtual method to generate the HTML statements defining a button control. Instead, create descendants of TQueryButton, such as TResetQueryButton (see page 2022) or TSubmitQueryButton (see page 2038).

See Also

TResetQueryButton (see page 2022)

TSubmitQueryButton (see page 2038)

IWebContent

TQueryForm (see page 1971)

37.1.33.1 TQueryButton Members

The following tables list the members exposed by TQueryButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1955)	This is DefaultCaption, a member of class TQueryButton.
	InputType (see page 1956)	This is InputType, a member of class TQueryButton.

Protected Methods

	Name	Description
	Content (see page 2060)	Returns the HTML that defines the button control.
V	EventContent (see page 1956)	This is EventContent, a member of class TQueryButton.
V	GetCaption (see page 1956)	Returns the value of the Caption property.
V	GetHTMLControlName (see page 1957)	This is GetHTMLControlName, a member of class TQueryButton.
	GetHTMLForm (see page 1957)	This is GetHTMLForm, a member of class TQueryButton.
V	GetInputType (see page 1957)	This is GetInputType, a member of class TQueryButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
	GetXMLComponent (see page 1957)	This is GetXMLComponent, a member of class TQueryButton.
V	ImplContent (see page 1958)	Provides the underlying implementation for the Content method.
V	Notification (see page 1958)	Forwards notification messages to all owned components.
	ParentXMLComponent (see page 1959)	This is ParentXMLComponent, a member of class TQueryButton.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetXMLComponent (see page 1959)	This is SetXMLComponent, a member of class TQueryButton.
	SetXMLUseParent (see page 1959)	This is SetXMLUseParent, a member of class TQueryButton.

Public Methods

	Name	Description
	Create (see page 1955)	Creates and initializes an instance of TQueryButton (see page 1953).

Published Properties

	Name	Description
	Caption (see page 1959)	Specifies the caption that appears on the button.
	Custom (see page 1960)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1960)	Adds style attributes to the generated button control.
	StyleRule (see page 1961)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1961)	Associates the button with the query form whose display controls it influences.
	XMLUseParent (see page 1962)	Specifies whether the button is associated with its parent's query form.

37.1.33.2 TQueryButton.Create Constructor

Creates and initializes an instance of TQueryButton (see page 1953).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TQueryButton(TComponent * AOwner);
```

Description

Call Create to instantiate a dataset button at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

AOwner is the component that is responsible for freeing the query button. Typically, this is the Web module that contains the InterenetExpress page producer that uses the query button's generated HTML.

37.1.33.3 TQueryButton Fields

The fields of the TQueryButton class are listed here.

Protected Fields

	Name	Description
	DefaultCaption (see page 1955)	This is DefaultCaption, a member of class TQueryButton.
	InputType (see page 1956)	This is InputType, a member of class TQueryButton.

37.1.33.3.1 TQueryButton.DefaultCaption Field

Pascal

```
DefaultCaption: string;
```

C++

```
UnicodeString DefaultCaption;
```

Description

This is DefaultCaption, a member of class TQueryButton.

37.1.33.3.2 TQueryButton.InputType Field

Pascal

```
InputType: string;
```

C++

```
UnicodeString InputType;
```

Description

This is InputType, a member of class TQueryButton.

37.1.33.4 TQueryButton Methods

The methods of the TQueryButton class are listed here.

Protected Methods

	Name	Description
	EventContent (see page 1956)	This is EventContent, a member of class TQueryButton.
	GetCaption (see page 1956)	Returns the value of the Caption property.
	GetHTMLControlName (see page 1957)	This is GetHTMLControlName, a member of class TQueryButton.
	GetHTMLForm (see page 1957)	This is GetHTMLForm, a member of class TQueryButton.
	GetInputType (see page 1957)	This is GetInputType, a member of class TQueryButton.
	GetXMLComponent (see page 1957)	This is GetXMLComponent, a member of class TQueryButton.
	ImplContent (see page 1958)	Provides the underlying implementation for the Content method.
	Notification (see page 1958)	Forwards notification messages to all owned components.
	ParentXMLComponent (see page 1959)	This is ParentXMLComponent, a member of class TQueryButton.
	SetXMLComponent (see page 1959)	This is SetXMLComponent, a member of class TQueryButton.
	SetXMLUseParent (see page 1959)	This is SetXMLUseParent, a member of class TQueryButton.

37.1.33.4.1 TQueryButton.EventContent Method

Pascal

```
function EventContent(Options: TWebContentOptions): string; virtual;
```

C++

```
virtual __fastcall UnicodeString EventContent(TWebContentOptions Options);
```

Description

This is EventContent, a member of class TQueryButton.

37.1.33.4.2 TQueryButton.GetCaption Method

Returns the value of the Caption property.

Pascal

```
function GetCaption: string; override;
```

C++

```
virtual __fastcall UnicodeString GetCaption();
```

Description

GetCaption is the protected read implementation for the Caption property.

See Also

Caption (see page 2062)

SetCaption (see page 2061)

37.1.33.4.3 TQueryButton.GetHTMLControlName Method

Pascal

```
function GetHTMLControlName: string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetHTMLControlName();
```

Description

This is GetHTMLControlName, a member of class TQueryButton.

37.1.33.4.4 TQueryButton.GetHTMLForm Method

Pascal

```
function GetHTMLForm: IHTMLForm;
```

C++

```
__fastcall IHTMLForm GetHTMLForm();
```

Description

This is GetHTMLForm, a member of class TQueryButton.

37.1.33.4.5 TQueryButton.GetInputType Method

Pascal

```
function GetInputType: string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetInputType();
```

Description

This is GetInputType, a member of class TQueryButton.

37.1.33.4.6 TQueryButton.GetXMLComponent Method

Pascal

```
function GetXMLComponent: TComponent;
```

C++

```
__fastcall TComponent * GetXMLComponent();
```

Description

This is GetXMLComponent, a member of class TQueryButton.

37.1.33.4.7 TQueryButton.ImplContent Method

Provides the underlying implementation for the Content method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content method calls ImplContent to generate the HTML that defines the button control. TWebButton introduces ImplContent as an abstract or, in C++ terminology, pure virtual method. TWebButton descendants must override ImplContent to generate the definition of their particular button control.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the Web button should format the HTML control it generates. It implements the ILayoutWebContent interface.

ImplContent returns a string of HTML defining the button control.

See Also

Content (see page 2060)

ILayoutWebContent

37.1.33.4.8 TQueryButton.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

37.1.33.4.9 TQueryButton.ParentXMLComponent Method

Pascal

```
function ParentXMLComponent: TComponent;
```

C++

```
__fastcall TComponent * ParentXMLComponent();
```

Description

This is ParentXMLComponent, a member of class TQueryButton.

37.1.33.4.10 TQueryButton.SetXMLComponent Method

Pascal

```
procedure SetXMLComponent(const Value: TComponent);
```

C++

```
__fastcall SetXMLComponent(const TComponent * Value);
```

Description

This is SetXMLComponent, a member of class TQueryButton.

37.1.33.4.11 TQueryButton.SetXMLUseParent Method

Pascal

```
procedure SetXMLUseParent(const Value: Boolean);
```

C++

```
__fastcall SetXMLUseParent(const Boolean Value);
```

Description

This is SetXMLUseParent, a member of class TQueryButton.

37.1.33.5 TQueryButton Properties

The properties of the TQueryButton class are listed here.

Published Properties

	Name	Description
	Caption (see page 1959)	Specifies the caption that appears on the button.
	Custom (see page 1960)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1960)	Adds style attributes to the generated button control.
	StyleRule (see page 1961)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1961)	Associates the button with the query form whose display controls it influences.
	XMLUseParent (see page 1962)	Specifies whether the button is associated with its parent's query form.

37.1.33.5.1 TQueryButton.Caption Property

Specifies the caption that appears on the button.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the caption that appears on the generated HTML button. The value of Caption is used as the value of a VALUE parameter on the HTML INPUT tag generated to implement the button.

See Also

Custom (see page 2062)

37.1.33.5.2 TQueryButton.Custom Property

Represents a tag option and its value in the generated INPUT tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

Use Custom to represent one or more options that qualify the INPUT tag generated by the Web button. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to add an ALIGN option that aligns the caption and a TABINDEX tag to control tab order, set Custom to

```
ALIGN=TOP TABINDEX=2
```

See Also

Caption (see page 2062)

Style (see page 2063)

37.1.33.5.3 TQueryButton.Style Property

Adds style attributes to the generated button control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to add a STYLE option that qualifies the INPUT tag generated by the Web button. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated INPUT tag as the value of a STYLE option.

See Also

Custom (see page 2062)

StyleRule (see page 2063)

37.1.33.5.4 TQueryButton.StyleRule Property

Applies a style in the page producer's style sheet to the button.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated button control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the button's Style property.

See Also

Style (see page 2063)

Styles (see page 2272)

StylesFile (see page 2273)

Styles

StylesFile

37.1.33.5.5 TQueryButton.XMLComponent Property

Associates the button with the query form whose display controls it influences.

Pascal

```
property XMLComponent: TComponent;
```

C++

```
__property TComponent * XMLComponent;
```

Description

Set XMLComponent to specify the Web item that generates the form whose controls the query button influences. If this query form contains the TQueryButtons (see page 1962) object that formats the query button with its siblings, you need not set the XMLComponent property. Instead, setting the XMLUseParent (see page 1962) property to true (the default) sets XMLComponent to the Query form that contains the parent TQueryButtons (see page 1962) object.

See Also

XMLUseParent (see page 1962)

TQueryForm (see page 1971)

37.1.33.5.6 TQueryButton.XMLUseParent Property

Specifies whether the button is associated with its parent's query form.

Pascal

```
property XMLUseParent: Boolean;
```

C++

```
__property Boolean XMLUseParent;
```

Description

Use XMLUseParent to indicate how the query button determines the form whose controls it influences. When XMLUseParent is false, the XMLComponent (see page 1961) property must be set to indicate the query form. When XMLUseParent is true, the button influences controls on the query form that contains its parent TQueryButtons (see page 1962) component.

See Also

XMLComponent (see page 1961)

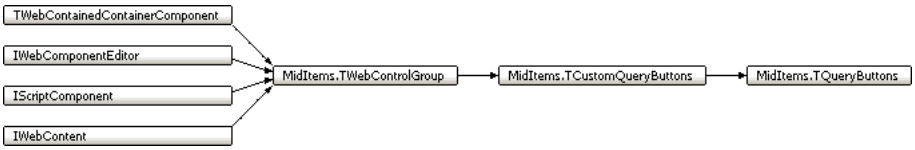
TQueryForm (see page 1971)

TQueryButtons (see page 1962)

37.1.34 TQueryButtons Class

TQueryButtons assembles the HTML generated by a set of TQueryButton (see page 1953) descendants.

Class Hierarchy



Pascal

```
TQueryButtons = class(TCustomQueryButtons);
```

C++

```
class TQueryButtons : public TCustomQueryButtons;
```

File

MidItems

Description

Add TQueryButtons as the child of a TQueryForm (see page 1971) component (or a layout group that is a child of a TQueryForm (see page 1971) component) to enable the generated HTML form to undo or submit changes made by end users. TQueryButtons generates HTML containing the controls generated by its children. These children must be descendants of TQueryButton (see page 1953), such as TSubmitQueryButton (see page 2038) and TResetQueryButton (see page 2022).

In order to let other Web items to use the HTML that defines the organized collection of controls, TQueryButtons inherits the IWebContent interface from TWebControlGroup (see page 2064).

Use the properties of TQueryButtons to customize the appearance of the group of buttons.

See Also

TQueryForm (see page 1971)

TResetQueryButton (see page 2022)

TSubmitQueryButton (see page 2038)

IWebContent

TQueryFieldGroup (see page 1965)

Using the Web Page Editor

37.1.34.1 TQueryButtons Members

The following tables list the members exposed by TQueryButtons.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	GetDefaultButtons (see page 1837)	This is GetDefaultButtons, a member of class TCustomQueryButtons.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
V	ImplAddElements (see page 1838)	Provides the underlying implementation for the AddElements method.
V	ImplAddElements (see page 2069)	Provides the underlying implementation for the AddElements (see page 2066) method.
V	ImplCanAddClass (see page 1838)	Provides the underlying implementation for the CanAddClass method.
V	ImplCanAddClass (see page 2070)	Provides the underlying implementation for the CanAddClass (see page 2067) method.
V	ImplContent (see page 1838)	Provides the underlying implementation for the Content method.
V	ImplContent (see page 2070)	Provides the underlying implementation for the Content (see page 2068) method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.
	VisibleButtons (see page 1839)	This is VisibleButtons, a member of class TCustomQueryButtons.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.

Public Methods

	Name	Description
V	Create (see page 2065)	Creates and initializes an instance of TWebControlGroup (see page 2064).
V	Destroy (see page 2066)	Frees an instance of TWebControlGroup (see page 2064).

Public Properties

	Name	Description
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.

Published Properties

	Name	Description
	Custom (see page 1964)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1964)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1965)	Applies a style in the page producer's style sheet to the group of buttons.

37.1.34.2 TQueryButtons Properties

The properties of the TQueryButtons class are listed here.

Published Properties

	Name	Description
	Custom (see page 1964)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1964)	Adds style attributes to the generated HTML table.
	StyleRule (see page 1965)	Applies a style in the page producer's style sheet to the group of buttons.

37.1.34.2.1 TQueryButtons.Custom Property

Represents a tag option and its value in the generated TABLE tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TCustomQueryButtons generates an HTML table to lay out the buttons listed in the WebFieldControls property. Use Custom to add options that qualify the TABLE tag. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to add a border around the query buttons and color the background blue, you can set Custom to

```
BORDER=2 BGCOLOR=BLUE
```

See Also

Style (see page 1840)

StyleRule (see page 1840)

WebFieldControls (see page 2072)

37.1.34.2.2 TQueryButtons.Style Property

Adds style attributes to the generated HTML table.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

TCustomQueryButtons generates an HTML table to lay out the buttons listed in the WebFieldControls property. Use Style to add a STYLE option that qualifies the TABLE tag. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated TABLE tag as the value of a STYLE option.

See Also

Custom (see page 1839)

StyleRule (see page 1840)

WebFieldControls (see page 2072)

37.1.34.2.3 TQueryButtons.StyleRule Property

Applies a style in the page producer's style sheet to the group of buttons.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

TCustomQueryButtons generates an HTML table to lay out the buttons listed in the WebFieldControls property. Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated table. The value of StyleRule specifies one of the user-defined style names defined in the InternetExpress page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the Style property.

See Also

Style (see page 1840)

Styles (see page 2272)

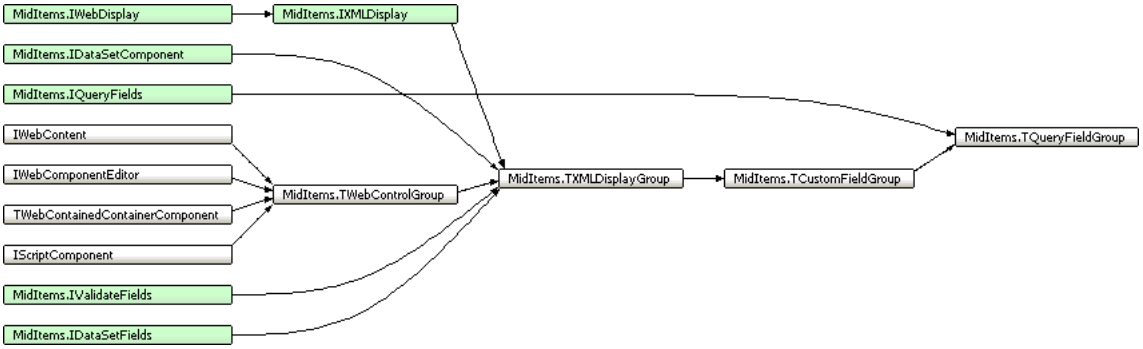
StylesFile (see page 2273)

WebFieldControls (see page 2072)

37.1.35 TQueryFieldGroup Class

TQueryFieldGroup arranges information on an HTML form as a set of labeled fields.

Class Hierarchy



Pascal

```
TQueryFieldGroup = class(TCustomFieldGroup, IQueryFields);
```

C++

```
class TQueryFieldGroup : public TCustomFieldGroup, public IQueryFields;
```

File

MidItems

Description

Add TQueryFieldGroup to a query form component (TQueryForm (see page 1971)) or a layout group (TLayoutGroup (see page 1939)) that is a child for a query form. TQueryFieldGroup generates the HTML to display information in labeled fields. By default, TQueryFieldGroup displays information about the parameters for an XML broker, but it can be used to display any application-defined named values.

TQueryFieldGroup maintains a list of Web items that generate the individual display fields. These Web items are restricted to the following classes: TQueryText (see page 2000), TQueryTextArea (see page 2010), TQuerySelectOptions (see page 1989), and TQueryRadioGroup (see page 1976). If the query field group displays parameter values, you can bind the query field group to an XML data packet or dataset field within an XML data packet. The Web items can then use that binding to determine the source of the parameters they display.

You can also use the properties of TQueryFieldGroup to affect the overall appearance of the set of fields.

See Also

TQueryForm (see page 1971)

TLayoutGroup (see page 1939)

TQueryRadioGroup (see page 1976)

TQuerySelectOptions (see page 1989)

TQueryText (see page 2000)

TQueryTextArea (see page 2010)

TXMLBroker

TWebDataDisplay (see page 2073)

Using the Web Page Editor

37.1.35.1 TQueryFieldGroup Members

The following tables list the members exposed by TQueryFieldGroup.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	CreateDefaultFields (see page 2179)	This is CreateDefaultFields, a member of class TXMLDisplayGroup.
	FindField (see page 2180)	This is FindField, a member of class TXMLDisplayGroup.
	FindParam (see page 1969)	This is FindParam, a member of class TQueryFieldGroup.
	GetDataSet (see page 2180)	Returns the value of the DataSet property.
	GetDataSetField (see page 2180)	This is GetDataSetField, a member of class TXMLDisplayGroup.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
	GetEnableValidateFields (see page 2180)	This is GetEnableValidateFields, a member of class TXMLDisplayGroup.
V	GetFieldsList (see page 2181)	This is GetFieldsList, a member of class TXMLDisplayGroup.
	GetIsMultipleRecordView (see page 2181)	This is GetIsMultipleRecordView, a member of class TXMLDisplayGroup.
V	GetLayoutAttributes (see page 1828)	Returns an object that specifies how the generated group of controls should look.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetParamsList (see page 1969)	This is GetParamsList, a member of class TQueryFieldGroup.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
	GetVisibleFields (see page 2181)	This is GetVisibleFields, a member of class TXMLDisplayGroup.
	GetXMLBroker (see page 2181)	Returns the value of the XMLBroker property.
	GetXMLDataSet (see page 2182)	Returns the value of the XMLDataSet property.
	GetXMLDisplayName (see page 2182)	This is GetXMLDisplayName, a member of class TXMLDisplayGroup.
	GetXMLRowSetName (see page 2182)	This is GetXMLRowSetName, a member of class TXMLDisplayGroup.
	HasStatusField (see page 2182)	This is HasStatusField, a member of class TXMLDisplayGroup.
V	ImplAddElements (see page 2069)	Provides the underlying implementation for the AddElements (see page 2066) method.
V	ImplAddElements (see page 2183)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 1969)	Provides the underlying implementation for the CanAddClass method.
V	ImplContent (see page 1828)	Provides the underlying implementation for the Content method.
V	ImplContent (see page 2070)	Provides the underlying implementation for the Content (see page 2068) method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	ImplGetSubComponents (see page 2183)	Provides the underlying implementation for the GetSubComponents method.
V	ImplIsMultipleRecordView (see page 2183)	This is ImplIsMultipleRecordView, a member of class TXMLDisplayGroup.
	IsFieldInUse (see page 2184)	This is IsFieldInUse, a member of class TXMLDisplayGroup.
	IsParamInUse (see page 1970)	This is IsParamInUse, a member of class TQueryFieldGroup.
V	Notification (see page 2184)	Forwards notification messages to all owned components.
	SetDataSetField (see page 2184)	This is SetDataSetField, a member of class TXMLDisplayGroup.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.
	SetEnableValidateFields (see page 2185)	This is SetEnableValidateFields, a member of class TXMLDisplayGroup.
	SetXMLBroker (see page 2185)	This is SetXMLBroker, a member of class TXMLDisplayGroup.
	ValidateDataSet (see page 2185)	This is ValidateDataSet, a member of class TXMLDisplayGroup.
	ValidateFields (see page 2185)	This is ValidateFields, a member of class TXMLDisplayGroup.
V	XMLDataChange (see page 2185)	This is XMLDataChange, a member of class TXMLDisplayGroup.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.
	EnableValidateFields (see page 2186)	This is EnableValidateFields, a member of class TXMLDisplayGroup.
R	XMLDataSet (see page 2187)	Indicates the TXMLDataSet object that represents the dataset to which the Web item is bound.

Public Methods

	Name	Description
V	Create (see page 1827)	Creates and initializes an instance of TCustomFieldGroup (see page 1824).
V	Create (see page 2065)	Creates and initializes an instance of TWebControlGroup (see page 2064).
V	Create (see page 2178)	Creates and initializes an instance of TXMLDisplayGroup (see page 2176).
V	Destroy (see page 1827)	Frees an instance of TCustomFieldGroup (see page 1824).
V	Destroy (see page 2066)	Frees an instance of TWebControlGroup (see page 2064).
V	Destroy (see page 2178)	Frees an instance of TXMLDisplayGroup (see page 2176).
	GetDataSet (see page 2201)	Returns the value of the DataSet (see page 2202) property.
	GetEnableValidateFields (see page 2225)	This is GetEnableValidateFields, a member of class IValidateFields.
	GetFieldsList (see page 2207)	This is GetFieldsList, a member of class IDataSetFields.
	GetIsMultipleRecordView (see page 2227)	This is GetIsMultipleRecordView, a member of class IWebDisplay.
	GetIsMultipleRecordView (see page 2229)	This is GetIsMultipleRecordView, a member of class IXMLDisplay.
	GetVisibleFields (see page 2207)	This is GetVisibleFields, a member of class IDataSetFields.
	GetXMLBroker (see page 2201)	Returns the value of the XMLBroker (see page 2203) property.
	GetXMLDataSet (see page 2202)	Returns the value of the XMLDataSet (see page 2203) property.
	GetXMLDisplayName (see page 2230)	This is GetXMLDisplayName, a member of class IXMLDisplay.
	GetXMLRowSetName (see page 2230)	This is GetXMLRowSetName, a member of class IXMLDisplay.
	HasStatusField (see page 2207)	This is HasStatusField, a member of class IDataSetFields.
	IsFieldInUse (see page 2207)	This is IsFieldInUse, a member of class IDataSetFields.
	SetEnableValidateFields (see page 2225)	This is SetEnableValidateFields, a member of class IValidateFields.
	ValidateDataSet (see page 2207)	This is ValidateDataSet, a member of class IDataSetFields.
	ValidateDataSet (see page 2221)	This is ValidateDataSet, a member of class IQueryFields.
	ValidateFields (see page 2225)	This is ValidateFields, a member of class IValidateFields.

Public Properties

	Name	Description
R	DataSet (see page 2202)	Provides access to an interface for working with the associated dataset.
	EnableValidateFields (see page 2226)	This is EnableValidateFields, a member of class IValidateFields.
R	IsMultipleRecordView (see page 2227)	
R	IsMultipleRecordView (see page 2230)	
R	VisibleFields (see page 2208)	This is VisibleFields, a member of class IDataSetFields.
R	VisibleFields (see page 2186)	Indicates the controls displayed by the HTML that the XML display group generates.
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.
R	XMLBroker (see page 2203)	Indicates the XML broker that fetches the data packets to which the Web item is bound.
R	XMLDataSet (see page 2203)	Indicates the TXMLDataSet (see page 2164) object that represents the dataset to which the Web item is bound.
R	XMLDisplayName (see page 2231)	This is XMLDisplayName, a member of class IXMLDisplay.
R	XMLRowSetName (see page 2231)	This is XMLRowSetName, a member of class IXMLDisplay.

Published Properties

	Name	Description
	Custom (see page 1970)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1971)	Adds style attributes to the generated set of fields.
	StyleRule (see page 1971)	Applies a style in the page producer's style sheet to the table.

	XMLBroker (see page 2187)	Binds the XML display group to an XML broker that represents a dataset.
	XMLDataSetField (see page 2188)	Binds the XML display group to a nested dataset from the XML data packet.

37.1.35.2 TQueryFieldGroup Methods

The methods of the TQueryFieldGroup class are listed here.

Protected Methods

	Name	Description
	FindParam (see page 1969)	This is FindParam, a member of class TQueryFieldGroup.
	GetParamsList (see page 1969)	This is GetParamsList, a member of class TQueryFieldGroup.
V	ImplCanAddClass (see page 1969)	Provides the underlying implementation for the CanAddClass method.
	IsParamInUse (see page 1970)	This is IsParamInUse, a member of class TQueryFieldGroup.

37.1.35.2.1 TQueryFieldGroup.FindParam Method

Pascal

```
function FindParam(const AName: string): TComponent;
```

C++

```
__fastcall TComponent * FindParam(const UnicodeString AName);
```

Description

This is FindParam, a member of class TQueryFieldGroup.

37.1.35.2.2 TQueryFieldGroup.GetParamsList Method

Pascal

```
procedure GetParamsList(List: TStrings);
```

C++

```
__fastcall GetParamsList(TStrings List);
```

Description

This is GetParamsList, a member of class TQueryFieldGroup.

37.1.35.2.3 TQueryFieldGroup.ImplCanAddClass Method

Provides the underlying implementation for the CanAddClass method.

Pascal

```
function ImplCanAddClass(AParent: TComponent; AClass: TClass): Boolean; override;
```

C++

```
virtual __fastcall Boolean ImplCanAddClass(TComponent * AParent, TClass AClass);
```

Description

The CanAddClass method calls ImpleCanAddClass to determine whether the TWebControlGroup descendant can add a component of type AClass to its WebComponents property list. AParent is the TWebControlGroup instance to which the component will be added.

In TWebControlGroup, ImplCanAddClass always returns true. TWebControlGroup descendants override ImplCanAddClass to

indicate what types of Web Items they can use when generating their HTML content.

See Also

CanAddClass (see page 2067)

WebFieldControls (see page 2072)

37.1.35.2.4 TQueryFieldGroup.IsParamInUse Method

Pascal

```
function IsParamInUse(AName: string): Boolean;
```

C++

```
__fastcall Boolean IsParamInUse(UnicodeString AName);
```

Description

This is IsParamInUse, a member of class TQueryFieldGroup.

37.1.35.3 TQueryFieldGroup Properties

The properties of the TQueryFieldGroup class are listed here.

Published Properties

	Name	Description
	Custom (see page 1970)	Represents a tag option and its value in the generated TABLE tag.
	Style (see page 1971)	Adds style attributes to the generated set of fields.
	StyleRule (see page 1971)	Applies a style in the page producer's style sheet to the table.

37.1.35.3.1 TQueryFieldGroup.Custom Property

Represents a tag option and its value in the generated TABLE tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TCustomFieldGroup generates an HTML table to manage the layout of the HTML controls that its Web items generate. Use Custom to add options that qualify the TABLE tag. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to surround the set of fields with a border and give them a blue background, set Custom to

```
BORDER=2 BGCOLOR=BLUE
```

See Also

Style (see page 1829)

StyleRule (see page 1830)

37.1.35.3.2 TQueryFieldGroup.Style Property

Adds style attributes to the generated set of fields.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

TCustomFieldGroup generates an HTML table to lay out the fields that its Web items generate. Use Style to add a STYLE option that qualifies the TABLE tag. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated TABLE tag as the value of a STYLE option.

See Also

Custom (see page 1829)

StyleRule (see page 1830)

37.1.35.3.3 TQueryFieldGroup.StyleRule Property

Applies a style in the page producer's style sheet to the table.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated set of fields. The value of StyleRule specifies one of the user-defined style names defined in the InternetExpress page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the Style property.

See Also

Style (see page 1829)

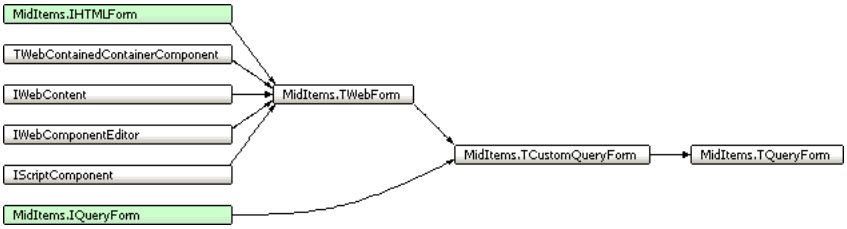
Styles (see page 2272)

StylesFile (see page 2273)

37.1.36 TQueryForm Class

TQueryForm generates HTML forms for letting users enter application-defined values.

Class Hierarchy



Pascal

```
TQueryForm = class(TCustomQueryForm);
```

C++

```
class TQueryForm : public TCustomQueryForm;
```

File

MidItems

Description

Add TQueryForm to the Web items of an InternetExpress page producer to add an HTML form that displays application-defined values that users can view and edit. Query forms submit these values to the InternetExpress application using a specified URL and method type.

TQueryForm generates the HTML FORM tag and assembles the HTML generated by its Web items. It restricts the Web items that generate the contents of the form to components that display application-defined values (TQueryFieldGroup (see page 1965)), components that contain buttons to operate on those displays (TQueryButtons (see page 1962)), and components that manage the layout of HTML controls (TLayoutGroup (see page 1939)). At design time, you can add these Web items to the form using the Web page editor.

See Also

TQueryButtons (see page 1962)

TBasePageItemsProducer (see page 2966)

TInetXPageProducer (see page 2285)

TQueryFieldGroup (see page 1965)

TLayoutGroup (see page 1939)

Using the Web Page Editor

37.1.36.1 TQueryForm Members

The following tables list the members exposed by TQueryForm.

Protected Methods

	Name	Description
	AddElements (see page 2105)	Used internally to generate any javascript declarations needed by the form.
	AssignStringsCallback (see page 1844)	This is AssignStringsCallback, a member of class TCustomQueryForm.
	CanAddClass (see page 2105)	Specifies whether a Web item of a specified type can be added to the WebComponents (see page 2112) property list.
	Content (see page 2106)	Returns the HTML that defines the HTML form.

	GetDefaultWebComponents (see page 2106)	Returns nil (Delphi) or NULL (C++).
	GetHTMLFormName (see page 2107)	Returns the name of the generated HTML form.
	GetHTMLFormTag (see page 1844)	This is GetHTMLFormTag, a member of class TCustomQueryForm.
	GetHTMLFormVarName (see page 2107)	Returns the name of the javascript variable that refers to the generated form.
	GetLayoutAttributes (see page 2107)	Returns the object that specifies how the generated form should look.
	GetSubComponents (see page 2108)	Returns the value of the WebComponents (see page 2112) property.
	GetVisibleFields (see page 2108)	Returns the value of the VisibleFields (see page 2112) property.
	ImplAddElements (see page 1844)	Provides the underlying implementation for the AddElements method.
	ImplAddElements (see page 2108)	Provides the underlying implementation for the AddElements (see page 2105) method.
	ImplCanAddClass (see page 1845)	Provides the underlying implementation for the CanAddClass method.
	ImplCanAddClass (see page 2109)	Provides the underlying implementation for the CanAddClass (see page 2105) method.
	ImplContent (see page 1845)	Provides the underlying implementation for the Content method.
	ImplContent (see page 2109)	Provides the underlying implementation for the Content (see page 2106) method.
	MethodString (see page 1846)	This is MethodString, a member of class TCustomQueryForm.
	SetDefaultWebComponents (see page 2110)	Prevents an application from trying to assign a default set of Web items on the form.

Protected Properties

	Name	Description
	VisibleFields (see page 2112)	Lists the Web items whose generated content appears in the HTML form.

Public Methods

	Name	Description
	AssignStrings (see page 1843)	Assigns values to display controls from a list of name/value pairs.
	Create (see page 1843)	Creates and initializes an instance of TCustomQueryForm (see page 1841).
	Create (see page 2104)	Creates and initializes an instance of TWebForm (see page 2102).
	Destroy (see page 2104)	Frees an instance of TWebForm (see page 2102).
	GetHTMLFormName (see page 2212)	Returns the name of the generated HTML form.
	GetHTMLFormVarName (see page 2212)	Returns the name of the javascript variable that refers to the generated form.

Public Properties

	Name	Description
	HTMLFormName (see page 2213)	Indicates the name of the generated HTML form.
	HTMLFormVarName (see page 2213)	Indicates the name of the javascript variable that represents the HTML form.
	WebComponents (see page 2112)	Maintains a list of components that generate HTML content for the TWebForm (see page 2102) descendant.

Published Properties

	Name	Description
	Action (see page 1974)	Specifies the path info portion of the URL to which the query submits request messages.
	Custom (see page 1974)	Represents a tag option and its value in the generated FORM tag.
	Method (see page 1975)	Specifies the METHOD header of messages that the query form submits.
	Style (see page 1975)	Adds style attributes to the generated form.
	StyleRule (see page 1975)	Applies a style in the page producer's style sheet to the button.

37.1.36.2 TQueryForm Properties

The properties of the TQueryForm class are listed here.

Published Properties

	Name	Description
	Action (see page 1974)	Specifies the path info portion of the URL to which the query submits request messages.
	Custom (see page 1974)	Represents a tag option and its value in the generated FORM tag.
	Method (see page 1975)	Specifies the METHOD header of messages that the query form submits.
	Style (see page 1975)	Adds style attributes to the generated form.
	StyleRule (see page 1975)	Applies a style in the page producer's style sheet to the button.

37.1.36.2.1 TQueryForm.Action Property

Specifies the path info portion of the URL to which the query submits request messages.

Pascal

```
property Action: string;
```

C++

```
__property UnicodeString Action;
```

Description

Use Action to specify the target URL for messages that the query form submits. Action supplies the PathInfo property of Web request messages from the query form. This value is matched to the PathInfo property of a Web action item when determining what object responds to those messages.

See Also

TCustomQueryForm (see page 1841)

PathInfo

PathInfo

37.1.36.2.2 TQueryForm.Custom Property

Represents a tag option and its value in the generated FORM tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

Use Custom to represent one or more options that qualify the FORM tag generated by the TWebForm descendant. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to specify that the form includes files with the text it submits and indicate that the page returned after submitting the form goes into a separate window, use a Custom value such as the following:

```
ENCTYPE="multipart/form-data" TARGET="ResultFrame"
```

See Also

Style (see page 2111)

37.1.36.2.3 TQueryForm.Method Property

Specifies the METHOD header of messages that the query form submits.

Pascal

```
property Method: TFormMethod;
```

C++

```
__property TFormMethod Method;
```

Description

Set Method to specify the METHOD header for messages that the query form submits. This value gives rise to the MethodType property of the HTTP request message. By matching this to the MethodType property of Web action items, a Web dispatcher determines what objects handle these messages.

See Also

TCustomQueryForm (see page 1841)

PathInfo

PathInfo

37.1.36.2.4 TQueryForm.Style Property

Adds style attributes to the generated form.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to add a STYLE option that qualifies the FORM tag generated by the TWebForm descendant. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated FORM tag as the value of a STYLE option.

See Also

Custom (see page 2110)

StyleRule (see page 2111)

37.1.36.2.5 TQueryForm.StyleRule Property

Applies a style in the page producer's style sheet to the button.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated form. The value of StyleRule specifies one of the user-defined style names defined in the InternetExpress page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the Style property.

See Also

Style (see page 2111)

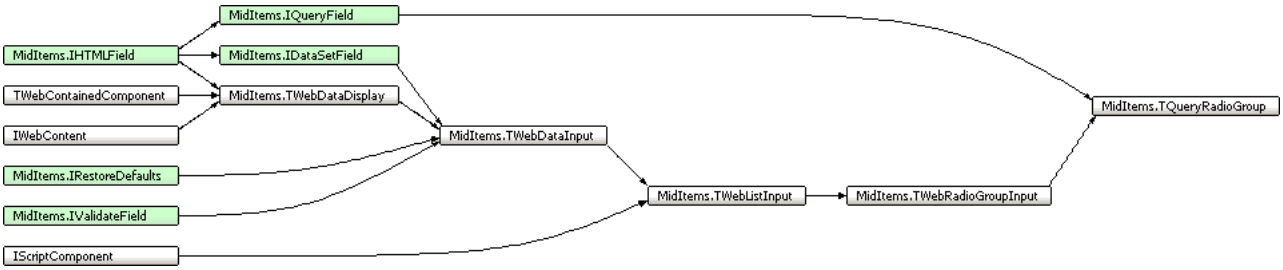
Styles (see page 2272)

StylesFile (see page 2273)

37.1.37 TQueryRadioGroup Class

TQueryRadioGroup creates an HTML radio group control to represent named parameter.

Class Hierarchy



Pascal

```
TQueryRadioGroup = class(TWebRadioGroupInput, IQueryField);
```

C++

```
class TQueryRadioGroup : public TWebRadioGroupInput, public IQueryField;
```

File

MidItems

Description

Add TQueryRadioGroup as the child of a TQueryFieldGroup (see page 1965) component to add a radio group control that represents a parameter value. By default, the children of TQueryFieldGroup (see page 1965) represent the parameters of an XML broker. However, TQueryRadioGroup can represent any named value that the InternetExpress application uses.

Use the properties and methods of TQueryRadioGroup to

Specify the field or parameter whose value the radio group displays.

Specify the possible values from which the user can choose.

Initialize the value of the radio group.

Specify a caption that describes the value of the radio group.

Customize the appearance of the radio group.

Fit the radio group into the tab order of the generated HTML form.

See Also

TQueryFieldGroup (see page 1965)

TXMLDisplayGroup (see page 2176)

TXMLBroker

37.1.37.1 TQueryRadioGroup Members

The following tables list the members exposed by TQueryRadioGroup.

Protected Methods

	Name	Description
	AddAttributes (see page 1979)	This is AddAttributes, a member of class TQueryRadioGroup.
	AddElements (see page 2117)	This is AddElements, a member of class TWebListInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
	ControlContent (see page 2117)	Returns the HTML that defines the display control (without its caption).
	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
	EventContent (see page 2126)	This is EventContent, a member of class TWebRadioGroupInput.
	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
	FormatInputs (see page 2117)	This is FormatInputs, a member of class TWebListInput.
	FormatInputs (see page 2126)	This is FormatInputs, a member of class TWebRadioGroupInput.
	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetCheckIndex (see page 1979)	This is GetCheckIndex, a member of class TQueryRadioGroup.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetSubComponents (see page 2118)	This is GetSubComponents, a member of class TWebListInput.
	GetText (see page 1980)	This is GetText, a member of class TQueryRadioGroup.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
	ImplAddElements (see page 2118)	This is ImplAddElements, a member of class TWebListInput.
	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
	ImplGetRowSetFieldAttributes (see page 2126)	This is ImplGetRowSetFieldAttributes, a member of class TWebRadioGroupInput.
	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
	LabelContent (see page 2082)	Returns the HTML for the display control's caption.

	Notification (see page 2118)	Forwards notification messages to all owned components.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
	RestoreFieldDefaults (see page 2126)	This is RestoreFieldDefaults, a member of class TWebRadioGroupInput.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetDataSet (see page 2119)	This is SetDataSet, a member of class TWebListInput.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetItems (see page 2119)	This is SetItems, a member of class TWebListInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	SetText (see page 1980)	This is SetText, a member of class TQueryRadioGroup.
	SetValues (see page 2119)	This is SetValues, a member of class TWebListInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.

Public Methods

	Name	Description
	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
	Create (see page 2116)	Creates and initializes an instance of TWebListInput (see page 2113).
	Create (see page 2125)	Creates and initializes an instance of TWebRadioGroupInput (see page 2122).
	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	Destroy (see page 2116)	Frees an instance of TWebListInput (see page 2113).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetParamName (see page 2218)	This is GetParamName, a member of class IQueryField.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 1980)	Indicates whether the component generates a control that can appear in a query form.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	SetParamName (see page 2218)	This is SetParamName, a member of class IQueryField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

Published Properties

	Name	Description
	Caption (see page 1981)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1981)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1982)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1982)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 1982)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayColumns (see page 1983)	Specifies the number of radio buttons that appear in each row of the control.
	DisplayWidth (see page 1983)	Specifies the width of the radio group, in characters.
	FieldName (see page 1984)	Specifies the name of the field whose value the generated control represents.
	Items (see page 1984)	Lists the possible selections that appear in the generated control.

	ItemsField (see page 1985)	Indicates which field in DataSet supplies the possible selections for the generated control.
	ParamName (see page 1985)	Specifies the name of the parameter whose value the generated control represents.
	ReadOnly (see page 1986)	Indicates whether end users can use the generated control to set values.
	Style (see page 1986)	Adds style attributes to the generated display control.
	StyleRule (see page 1986)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1987)	Determines the position of the generated control in the HTML form's tab order.
	Text (see page 1987)	Specifies the initial value of the radio group.
	Values (see page 1988)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 1988)	Indicates which field in DataSet supplies the possible values for the generated control.

37.1.37.2 TQueryRadioGroup Methods

The methods of the TQueryRadioGroup class are listed here.

Protected Methods

	Name	Description
	AddAttributes (see page 1979)	This is AddAttributes, a member of class TQueryRadioGroup.
	GetCheckIndex (see page 1979)	This is GetCheckIndex, a member of class TQueryRadioGroup.
	GetText (see page 1980)	This is GetText, a member of class TQueryRadioGroup.
	SetText (see page 1980)	This is SetText, a member of class TQueryRadioGroup.

Public Methods

	Name	Description
	IsQueryField (see page 1980)	Indicates whether the component generates a control that can appear in a query form.

37.1.37.2.1 TQueryRadioGroup.AddAttributes Method

Pascal

```
procedure AddAttributes(var Attrs: string); override;
```

C++

```
virtual __fastcall AddAttributes(UnicodeString Attrs);
```

Description

This is AddAttributes, a member of class TQueryRadioGroup.

37.1.37.2.2 TQueryRadioGroup.GetCheckIndex Method

Pascal

```
function GetCheckIndex(ItemsStrings: TStrings; ValuesStrings: TStrings): Integer; override;
```

C++

```
virtual __fastcall int GetCheckIndex(TStrings ItemsStrings, TStrings ValuesStrings);
```

Description

This is GetCheckIndex, a member of class TQueryRadioGroup.

37.1.37.2.3 TQueryRadioGroup.GetText Method

Pascal

```
function GetText: string;
```

C++

```
__fastcall UnicodeString GetText();
```

Description

This is GetText, a member of class TQueryRadioGroup.

37.1.37.2.4 TQueryRadioGroup.IsQueryField Method

Indicates whether the component generates a control that can appear in a query form.

Pascal

```
class function IsQueryField: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsQueryField();
```

Description

Call IsQueryField to determine whether a Web display component generates a control that represents a named value in a query form (TQueryForm (see page 1971)). IsQueryField returns true, because TQueryRadioGroup (see page 1976) is designed specifically for use by TQueryFieldGroup (see page 1965).

See Also

IsColumn (see page 2081)

TQueryFieldGroup (see page 1965)

TQueryForm (see page 1971)

37.1.37.2.5 TQueryRadioGroup.SetText Method

Pascal

```
procedure SetText(const Value: string);
```

C++

```
__fastcall SetText(const UnicodeString Value);
```

Description

This is SetText, a member of class TQueryRadioGroup.

37.1.37.3 TQueryRadioGroup Properties

The properties of the TQueryRadioGroup class are listed here.

Published Properties

	Name	Description
	Caption (see page 1981)	Specifies the text of a label that indicates what information the generated control displays.

	CaptionAttributes (see page 1981)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1982)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1982)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 1982)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayColumns (see page 1983)	Specifies the number of radio buttons that appear in each row of the control.
	DisplayWidth (see page 1983)	Specifies the width of the radio group, in characters.
	FieldName (see page 1984)	Specifies the name of the field whose value the generated control represents.
	Items (see page 1984)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 1985)	Indicates which field in DataSet supplies the possible selections for the generated control.
	ParamName (see page 1985)	Specifies the name of the parameter whose value the generated control represents.
	ReadOnly (see page 1986)	Indicates whether end users can use the generated control to set values.
	Style (see page 1986)	Adds style attributes to the generated display control.
	StyleRule (see page 1986)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1987)	Determines the position of the generated control in the HTML form's tab order.
	Text (see page 1987)	Specifies the initial value of the radio group.
	Values (see page 1988)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 1988)	Indicates which field in DataSet supplies the possible values for the generated control.

37.1.37.3.1 TQueryRadioGroup.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.37.3.2 TQueryRadioGroup.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.37.3.3 TQueryRadioGroup.CaptionPosition Property

Specifies where the label appears relative to the generated display control.

Pascal

```
property CaptionPosition: TCaptionPosition;
```

C++

```
__property TCaptionPosition CaptionPosition;
```

Description

Use CaptionPosition to indicate where the label should appear relative to the control that displays information.

Note: CaptionPosition has no effect on column controls. The position of column labels is always in the table header.

See Also

CaptionAttributes (see page 2084)

Caption (see page 2083)

37.1.37.3.4 TQueryRadioGroup.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.37.3.5 TQueryRadioGroup.DataSet Property

Specifies a dataset that supplies the possible values listed in the generated control.

Pascal

```
property DataSet: TDataSet;
```

C++

```
__property TDataSet DataSet;
```

Description

Use DataSet to specify a dataset from which the component can extract the selections that appear to the user and the underlying values they represent. The ItemsField property specifies the field in this dataset that lists the selections that appear in the generated control. The ValuesField property specifies the field in this dataset that lists the corresponding values that are assigned to the control.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

ItemsField (see page 2120)

ValuesField (see page 2121)

Items (see page 2120)

Values (see page 2121)

37.1.37.3.6 TQueryRadioGroup.DisplayColumns Property

Specifies the number of radio buttons that appear in each row of the control.

Pascal

```
property DisplayColumns: Integer;
```

C++

```
__property int DisplayColumns;
```

Description

Use DisplayColumns to specify how many radio buttons can appear in a single line of the radio group. If DisplayColumns is less than the number of items, the radio group displays its buttons in multiple rows.

See Also

ItemsField (see page 2120)

Items (see page 2120)

DisplayWidth (see page 2127)

37.1.37.3.7 TQueryRadioGroup.DisplayWidth Property

Specifies the width of the radio group, in characters.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

Description

Use DisplayWidth to limit the width of the generated control. TWebRadioGroup generates an INPUT tag of type RADIO to implement the radio group. DisplayWidth is added as a WIDTH option.

See Also

DisplayColumns (see page 2127)

Custom (see page 2084)

37.1.37.3.8 TQueryRadioGroup.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.37.3.9 TQueryRadioGroup.Items Property

Lists the possible selections that appear in the generated control.

Pascal

```
property Items: TStrings;
```

C++

```
__property TStrings Items;
```

Description

Use Items to supply a list of possible selections to the generated control. The end-user can choose from these selections to assign a value to the generated control. The value that is assigned comes from the corresponding entry in the Values property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items and Values properties, use the DataSet, ItemsField, and ValuesField properties.

See Also

ItemsField (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.37.3.10 TQueryRadioGroup.ItemsField Property

Indicates which field in DataSet supplies the possible selections for the generated control.

Pascal

```
property ItemsField: string;
```

C++

```
__property UnicodeString ItemsField;
```

Description

Use ItemsField to specify which field in the dataset specified by the DataSet property contains selections that appear in the generated control. The end-user can choose from these selections to assign a value to the generated control. The value that is assigned comes from the field specified by the ValuesField property.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.37.3.11 TQueryRadioGroup.ParamName Property

Specifies the name of the parameter whose value the generated control represents.

Pascal

```
property ParamName: string;
```

C++

```
__property UnicodeString ParamName;__property UnicodeString ParamName;
```

Description

Use ParamName to bind the Web data input component to a parameter in the XML data packet or supply an application-specific name to the value for this control. ParamName is only used when the IsQueryField method returns true. By default, the controls in a query form represent parameter values. However, they can be used to represent any named value.

If ParamName identifies one of the parameters of an XML broker, the XML broker property of the parent Web item identifies the broker that uses the parameter.

See Also

FieldName (see page 2094)

IsQueryField (see page 2081)

Params

XMLBroker (see page 2187)

Setting Web Item Properties

37.1.37.3.12 TQueryRadioGroup.ReadOnly Property

Indicates whether end users can use the generated control to set values.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use ReadOnly to indicate whether users can change the value displayed by the radio group. TWebRadioGroup generates an INPUT tag of type RADIO to implement the radio group. ReadOnly is added as a READONLY option.

See Also

DisplayWidth (see page 2127)

Custom (see page 2084)

37.1.37.3.13 TQueryRadioGroup.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.37.3.14 TQueryRadioGroup.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.37.3.15 TQueryRadioGroup.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.37.3.16 TQueryRadioGroup.Text Property

Specifies the initial value of the radio group.

Pascal

```
property Text: string;
```

C++

```
__property UnicodeString Text;
```

Description

Use Text to initialize the value that appears in the generated radio group. This should be one of the values listed in the Items (see page 1984) property or in the field referenced by the ItemsField (see page 1985) property.

See Also

Items (see page 2120)

ItemsField (see page 2120)

37.1.37.3.17 TQueryRadioGroup.Values Property

Lists the possible values that can be assigned to the control.

Pascal

```
property Values: TStrings;
```

C++

```
__property TStrings Values;
```

Description

Use Values to supply a list of possible values for the generated control. When the end-user chooses one of the selections listed by the Items property, the control is assigned the corresponding value in the Values property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items and Values properties, use the DataSet, ItemsField, and ValuesField properties.

See Also

Items (see page 2120)

ItemsField (see page 2120)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.37.3.18 TQueryRadioGroup.ValuesField Property

Indicates which field in DataSet supplies the possible values for the generated control.

Pascal

```
property ValuesField: string;
```

C++

```
__property UnicodeString ValuesField;
```

Description

Use ValuesField to specify which field in the dataset specified by the DataSet property contains values that can be assigned to the generated control. When the end-user selects the corresponding value form the ItemsField field, the control gets the value in the ValuesField field.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

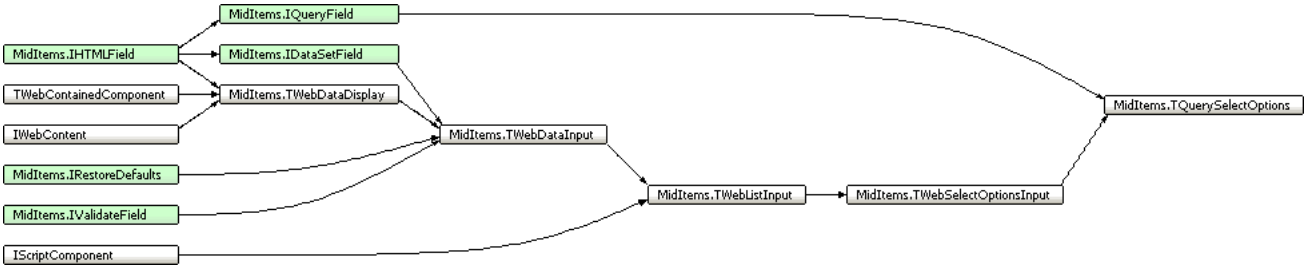
DataSet (see page 2119)

ItemsField (see page 2120)

37.1.38 TQuerySelectOptions Class

TQuerySelectOptions creates an HTML list control to represent a named parameter.

Class Hierarchy



Pascal

```
TQuerySelectOptions = class(TWebSelectOptionsInput, IQueryField);
```

C++

```
class TQuerySelectOptions : public TWebSelectOptionsInput, public IQueryField;
```

File

MidItems

Description

Add TQuerySelectOptions as the child of a TQueryFieldGroup (see page 1965) component to add a list control that represents a parameter value. By default, the children of TQueryFieldGroup (see page 1965) represent the parameters of an XML broker. However, TQuerySelectOptions can represent any named value that the InternetExpress application uses.

Use the properties and methods of TQuerySelectOptions to

Specify the field or parameter whose value the list box displays.

Specify the possible values from which the user can choose.

Assign an initial value to the list box.

Specify a caption that describes the value of the list box.

Customize the appearance of the list box.

Fit the list box into the tab order of the generated HTML form.

See Also

TQueryFieldGroup (see page 1965)

TXMLDisplayGroup (see page 2176)

TXMLBroker

37.1.38.1 TQuerySelectOptions Members

The following tables list the members exposed by TQuerySelectOptions.

Protected Methods

	Name	Description
V	AddAttributes (see page 2131)	This is AddAttributes, a member of class TWebSelectOptionsInput.
	AddElements (see page 2117)	This is AddElements, a member of class TWebListInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
A	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
V	ControlContent (see page 2117)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
V	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
V	EventContent (see page 2132)	This is EventContent, a member of class TWebSelectOptionsInput.
V	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
A	FormatInputs (see page 2117)	This is FormatInputs, a member of class TWebListInput.
V	FormatInputs (see page 2132)	This is FormatInputs, a member of class TWebSelectOptionsInput.
V	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
V	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
V	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
V	GetSelectIndex (see page 1992)	This is GetSelectIndex, a member of class TQuerySelectOptions.
	GetSubComponents (see page 2118)	This is GetSubComponents, a member of class TWebListInput.
	GetText (see page 1992)	This is GetText, a member of class TQuerySelectOptions.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplAddElements (see page 2118)	This is ImplAddElements, a member of class TWebListInput.
V	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
V	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
V	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
V	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
V	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
V	Notification (see page 2118)	Forwards notification messages to all owned components.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
V	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
V	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
V	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetDataSet (see page 2119)	This is SetDataSet, a member of class TWebListInput.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetItems (see page 2119)	This is SetItems, a member of class TWebListInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	SetText (see page 1993)	This is SetText, a member of class TQuerySelectOptions.
	SetValues (see page 2119)	This is SetValues, a member of class TWebListInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.

Public Methods

	Name	Description
	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
	Create (see page 2116)	Creates and initializes an instance of TWebListInput (see page 2113).
	Create (see page 2131)	Creates and initializes an instance of TWebSelectOptionsInput (see page 2128).
	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	Destroy (see page 2116)	Frees an instance of TWebListInput (see page 2113).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetParamName (see page 2218)	This is GetParamName, a member of class IQueryField.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 1992)	Indicates whether the component generates a control that can appear in a query form.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	SetParamName (see page 2218)	This is SetParamName, a member of class IQueryField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

Published Properties

	Name	Description
	Caption (see page 1993)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1994)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1994)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1994)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 1995)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayRows (see page 1995)	Specifies the number of rows that appear in the generated list box.
	FieldName (see page 1996)	Specifies the name of the field whose value the generated control represents.
	Items (see page 1996)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 1997)	Indicates which field in DataSet supplies the possible selections for the generated control.
	ParamName (see page 1997)	Specifies the name of the parameter whose value the generated control represents.
	Style (see page 1998)	Adds style attributes to the generated display control.
	StyleRule (see page 1998)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1999)	Determines the position of the generated control in the HTML form's tab order.
	Text (see page 1999)	Specifies the initial value selected in the list box.
	Values (see page 1999)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 2000)	Indicates which field in DataSet supplies the possible values for the generated control.

37.1.38.2 TQuerySelectOptions Methods

The methods of the TQuerySelectOptions class are listed here.

Protected Methods

	Name	Description
	GetSelectIndex (see page 1992)	This is GetSelectIndex, a member of class TQuerySelectOptions.
	GetText (see page 1992)	This is GetText, a member of class TQuerySelectOptions.
	SetText (see page 1993)	This is SetText, a member of class TQuerySelectOptions.

Public Methods

	Name	Description
	IsQueryField (see page 1992)	Indicates whether the component generates a control that can appear in a query form.

37.1.38.2.1 TQuerySelectOptions.GetSelectIndex Method

Pascal

```
function GetSelectIndex(ItemsStrings: TStrings; ValuesStrings: TStrings): Integer; override;
```

C++

```
virtual __fastcall int GetSelectIndex(TStrings ItemsStrings, TStrings ValuesStrings);
```

Description

This is GetSelectIndex, a member of class TQuerySelectOptions.

37.1.38.2.2 TQuerySelectOptions.GetText Method

Pascal

```
function GetText: string;
```

C++

```
__fastcall UnicodeString GetText();
```

Description

This is GetText, a member of class TQuerySelectOptions.

37.1.38.2.3 TQuerySelectOptions.IsQueryField Method

Indicates whether the component generates a control that can appear in a query form.

Pascal

```
class function IsQueryField: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsQueryField();
```

Description

Call IsQueryField to determine whether a Web display component generates a control that represents a named value in a query form (TQueryForm (see page 1971)). IsQueryField returns true, because TQuerySelectOptions (see page 1989) is designed specifically for use by TQueryFieldGroup (see page 1965).

See Also

IsColumn (see page 2081)

TQueryFieldGroup (see page 1965)

TQueryForm (see page 1971)

37.1.38.2.4 TQuerySelectOptions.SetText Method

Pascal

```
procedure SetText(const Value: string);
```

C++

```
__fastcall SetText(const UnicodeString Value);
```

Description

This is SetText, a member of class TQuerySelectOptions.

37.1.38.3 TQuerySelectOptions Properties

The properties of the TQuerySelectOptions class are listed here.

Published Properties

	Name	Description
	Caption (see page 1993)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 1994)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 1994)	Specifies where the label appears relative to the generated display control.
	Custom (see page 1994)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 1995)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayRows (see page 1995)	Specifies the number of rows that appear in the generated list box.
	FieldName (see page 1996)	Specifies the name of the field whose value the generated control represents.
	Items (see page 1996)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 1997)	Indicates which field in DataSet supplies the possible selections for the generated control.
	ParamName (see page 1997)	Specifies the name of the parameter whose value the generated control represents.
	Style (see page 1998)	Adds style attributes to the generated display control.
	StyleRule (see page 1998)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 1999)	Determines the position of the generated control in the HTML form's tab order.
	Text (see page 1999)	Specifies the initial value selected in the list box.
	Values (see page 1999)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 2000)	Indicates which field in DataSet supplies the possible values for the generated control.

37.1.38.3.1 TQuerySelectOptions.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.38.3.2 TQuerySelectOptions.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.38.3.3 TQuerySelectOptions.CaptionPosition Property

Specifies where the label appears relative to the generated display control.

Pascal

```
property CaptionPosition: TCaptionPosition;
```

C++

```
__property TCaptionPosition CaptionPosition;
```

Description

Use CaptionPosition to indicate where the label should appear relative to the control that displays information.

Note: CaptionPosition has no effect on column controls. The position of column labels is always in the table header.

See Also

CaptionAttributes (see page 2084)

Caption (see page 2083)

37.1.38.3.4 TQuerySelectOptions.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.38.3.5 TQuerySelectOptions.DataSet Property

Specifies a dataset that supplies the possible values listed in the generated control.

Pascal

```
property DataSet: TDataSet;
```

C++

```
__property TDataSet DataSet;
```

Description

Use DataSet to specify a dataset from which the component can extract the selections that appear to the user and the underlying values they represent. The ItemsField property specifies the field in this dataset that lists the selections that appear in the generated control. The ValuesField property specifies the field in this dataset that lists the corresponding values that are assigned to the control.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

ItemsField (see page 2120)

ValuesField (see page 2121)

Items (see page 2120)

Values (see page 2121)

37.1.38.3.6 TQuerySelectOptions.DisplayRows Property

Specifies the number of rows that appear in the generated list box.

Pascal

```
property DisplayRows: Integer;
```

C++

```
__property int DisplayRows;
```

Description

Use DisplayRows to specify how many rows appear in the generated list control. If DisplayRows is less than the number of items, the generated list displays only DisplayRows options at one time, but can scroll to display the hidden items. If DisplayRows is 1, the generated control becomes a drop-down list box.

See Also

ItemsField (see page 2120)

Items (see page 2120)

37.1.38.3.7 TQuerySelectOptions.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.38.3.8 TQuerySelectOptions.Items Property

Lists the possible selections that appear in the generated control.

Pascal

```
property Items: TStrings;
```

C++

```
__property TStrings Items;
```

Description

Use Items to supply a list of possible selections to the generated control. The end-user can choose from these selections to

assign a value to the generated control. The value that is assigned comes from the corresponding entry in the Values property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items and Values properties, use the DataSet, ItemsField, and ValuesField properties.

See Also

ItemsField (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.38.3.9 TQuerySelectOptions.ItemsField Property

Indicates which field in DataSet supplies the possible selections for the generated control.

Pascal

```
property ItemsField: string;
```

C++

```
__property UnicodeString ItemsField;
```

Description

Use ItemsField to specify which field in the dataset specified by the DataSet property contains selections that appear in the generated control. The end-user can choose from these selections to assign a value to the generated control. The value that is assigned comes from the field specified by the ValuesField property.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.38.3.10 TQuerySelectOptions.ParamName Property

Specifies the name of the parameter whose value the generated control represents.

Pascal

```
property ParamName: string;
```

C++

```
__property UnicodeString ParamName;__property UnicodeString ParamName;
```

Description

Use ParamName to bind the Web data input component to a parameter in the XML data packet or supply an application-specific name to the value for this control. ParamName is only used when the IsQueryField method returns true. By default, the controls in a query form represent parameter values. However, they can be used to represent any named value.

If ParamName identifies one of the parameters of an XML broker, the XML broker property of the parent Web item identifies the

broker that uses the parameter.

See Also

FieldName (see page 2094)

IsQueryField (see page 2081)

Params

XMLBroker (see page 2187)

Setting Web Item Properties

37.1.38.3.11 TQuerySelectOptions.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.38.3.12 TQuerySelectOptions.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.38.3.13 TQuerySelectOptions.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.38.3.14 TQuerySelectOptions.Text Property

Specifies the initial value selected in the list box.

Pascal

```
property Text: string;
```

C++

```
__property UnicodeString Text;
```

Description

Use Text to initialize the value of the generated list control. Text should specify one of the values listed in the Items (see page 1996) property or referenced by the ItemsField (see page 1997) property.

See Also

Items (see page 2120)

ItemsField (see page 2120)

37.1.38.3.15 TQuerySelectOptions.Values Property

Lists the possible values that can be assigned to the control.

Pascal

```
property Values: TStrings;
```

C++

```
__property TStrings Values;
```

Description

Use Values to supply a list of possible values for the generated control. When the end-user chooses one of the selections listed by the Items property, the control is assigned the corresponding value in the Values property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items and Values properties, use the DataSet, ItemsField, and ValuesField properties.

See Also

Items (see page 2120)

ItemsField (see page 2120)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.38.3.16 TQuerySelectOptions.ValuesField Property

Indicates which field in DataSet supplies the possible values for the generated control.

Pascal

```
property ValuesField: string;
```

C++

```
__property UnicodeString ValuesField;
```

Description

Use ValuesField to specify which field in the dataset specified by the DataSet property contains values that can be assigned to the generated control. When the end-user selects the corresponding value form the ItemsField field, the control gets the value in the ValuesField field.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

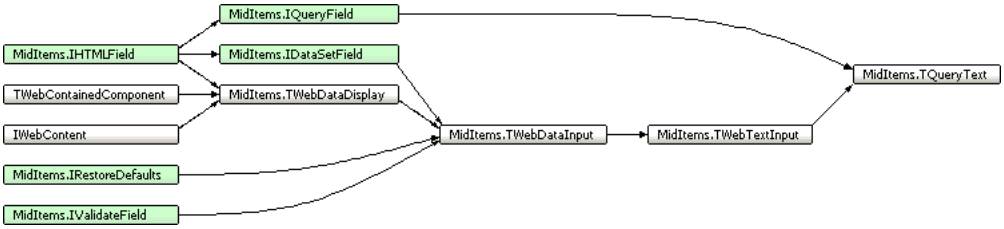
DataSet (see page 2119)

ItemsField (see page 2120)

37.1.39 TQueryText Class

TQueryText creates a single-line HTML input control to represent a named parameter.

Class Hierarchy



Pascal

```
TQueryText = class(TWebTextInput, IQueryField);
```

C++

```
class TQueryText : public TWebTextInput, public IQueryField;
```

File

MidItems

Description

Add TQueryText as the child of a TQueryFieldGroup (see page 1965) component to add an edit control that represents a parameter value. By default, the children of TQueryFieldGroup (see page 1965) represent the parameters of an XML broker. However, TQueryText can represent any named value that the InternetExpress application uses.

Use the properties and methods of TQueryText to

Specify the field or parameter whose value the control displays.

Assign an initial value to the control.

Specify a caption that describes the value of the control.

Customize the appearance of the input control.

Fit the control into the tab order of the generated HTML form.

See Also

TQueryFieldGroup (see page 1965)

TXMLDisplayGroup (see page 2176)

TXMLBroker

37.1.39.1 TQueryText Members

The following tables list the members exposed by TQueryText.

Protected Methods

	Name	Description
	AddAttributes (see page 2004)	This is AddAttributes, a member of class TQueryText.
	Content (see page 2076)	Returns the HTML that defines the display control.
	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
	ControlContent (see page 2148)	Returns the HTML that defines the display control (without its caption).
	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.

	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetText (see page 2004)	This is GetText, a member of class TQueryText.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
	ImplGetRowSetFieldAttributes (see page 2149)	This is ImplGetRowSetFieldAttributes, a member of class TWebTextInput.
	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
	RestoreFieldDefaults (see page 2149)	This is RestoreFieldDefaults, a member of class TWebTextInput.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	SetText (see page 2004)	This is SetText, a member of class TQueryText.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.

Public Methods

	Name	Description
	Create (see page 2003)	Creates and initializes an instance of TQueryText (see page 2000).
	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetParamName (see page 2218)	This is GetParamName, a member of class IQueryField.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 2004)	Indicates whether the component generates a control that can appear in a query form.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.

	SetParamName (see page 2218)	This is SetParamName, a member of class IQueryField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

Published Properties

	Name	Description
	Caption (see page 2005)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2005)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 2006)	Specifies where the label appears relative to the generated display control.
	Custom (see page 2006)	Represents a tag option and its value in the tag for the display control.
	DisplayWidth (see page 2007)	Specifies the horizontal size of the generated control.
	FieldName (see page 2007)	Specifies the name of the field whose value the generated control represents.
	MaxWidth (see page 2007)	Limits the length of text users can type into the control.
	ParamName (see page 2008)	Specifies the name of the parameter whose value the generated control represents.
	ReadOnly (see page 2008)	Indicates whether end users can use the generated control to set values.
	Style (see page 2009)	Adds style attributes to the generated display control.
	StyleRule (see page 2009)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2010)	Determines the position of the generated control in the HTML form's tab order.
	Text (see page 2010)	Specifies the initial value of the generated control.

37.1.39.2 TQueryText.Create Constructor

Creates and initializes an instance of TQueryText (see page 2000).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TQueryText(TComponent * AOwner);
```

Description

Call Create to instantiate TQueryText (see page 2000) at runtime. TQueryText (see page 2000) components added at design time using the Web Page editor are instantiated automatically.

37.1.39.3 TQueryText Methods

The methods of the TQueryText class are listed here.

Protected Methods

	Name	Description
V	AddAttributes (see page 2004)	This is AddAttributes, a member of class TQueryText.
	GetText (see page 2004)	This is GetText, a member of class TQueryText.
	SetText (see page 2004)	This is SetText, a member of class TQueryText.

Public Methods

	Name	Description
V	IsQueryField (see page 2004)	Indicates whether the component generates a control that can appear in a query form.

37.1.39.3.1 TQueryText.AddAttributes Method

Pascal

```
procedure AddAttributes(var Attrs: string); override;
```

C++

```
virtual __fastcall AddAttributes(UnicodeString Attrs);
```

Description

This is AddAttributes, a member of class TQueryText.

37.1.39.3.2 TQueryText.GetText Method

Pascal

```
function GetText: string;
```

C++

```
__fastcall UnicodeString GetText();
```

Description

This is GetText, a member of class TQueryText.

37.1.39.3.3 TQueryText.IsQueryField Method

Indicates whether the component generates a control that can appear in a query form.

Pascal

```
class function IsQueryField: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsQueryField();
```

Description

Call IsQueryField to determine whether a Web display component generates a control that represents a named value in a query form (TQueryForm (see page 1971)). IsQueryField returns true, because TQueryText (see page 2000) is designed specifically for TQueryFieldGroup (see page 1965) to use.

See Also

IsColumn (see page 2081)

TQueryFieldGroup (see page 1965)

TQueryForm (see page 1971)

37.1.39.3.4 TQueryText.SetText Method

Pascal

```
procedure SetText(const Value: string);
```

C++

```
__fastcall SetText(const UnicodeString Value);
```

Description

This is SetText, a member of class TQueryText.

37.1.39.4 TQueryText Properties

The properties of the TQueryText class are listed here.

Published Properties

	Name	Description
	Caption (see page 2005)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2005)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 2006)	Specifies where the label appears relative to the generated display control.
	Custom (see page 2006)	Represents a tag option and its value in the tag for the display control.
	DisplayWidth (see page 2007)	Specifies the horizontal size of the generated control.
	FieldName (see page 2007)	Specifies the name of the field whose value the generated control represents.
	MaxWidth (see page 2007)	Limits the length of text users can type into the control.
	ParamName (see page 2008)	Specifies the name of the parameter whose value the generated control represents.
	ReadOnly (see page 2008)	Indicates whether end users can use the generated control to set values.
	Style (see page 2009)	Adds style attributes to the generated display control.
	StyleRule (see page 2009)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2010)	Determines the position of the generated control in the HTML form's tab order.
	Text (see page 2010)	Specifies the initial value of the generated control.

37.1.39.4.1 TQueryText.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.39.4.2 TQueryText.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.39.4.3 TQueryText.CaptionPosition Property

Specifies where the label appears relative to the generated display control.

Pascal

```
property CaptionPosition: TCaptionPosition;
```

C++

```
__property TCaptionPosition CaptionPosition;
```

Description

Use CaptionPosition to indicate where the label should appear relative to the control that displays information.

Note: CaptionPosition has no effect on column controls. The position of column labels is always in the table header.

See Also

CaptionAttributes (see page 2084)

Caption (see page 2083)

37.1.39.4.4 TQueryText.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.39.4.5 TQueryText.DisplayWidth Property

Specifies the horizontal size of the generated control.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

Description

Use DisplayWidth to indicate the desired width of the generated HTML control. DisplayWidth specifies the number of characters that can be displayed in the control.

Note: DisplayWidth controls the size of the generated control, not the size of the text that users can enter. If the user enters a string longer than DisplayWidth characters, the text scrolls in the control.

37.1.39.4.6 TQueryText.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.39.4.7 TQueryText.MaxWidth Property

Limits the length of text users can type into the control.

Pascal

```
property MaxWidth: Integer;
```

C++

```
__property int MaxWidth;
```

Description

Use MaxWidth to limit the length of strings (in characters) that users can type into the generated control.

Note: MaxWidth has no effect on the size of the generated control. To limit the size of the control, use the DisplayWidth (see page 2007) property.

See Also

DisplayWidth (see page 2150)

37.1.39.4.8 TQueryText.ParamName Property

Specifies the name of the parameter whose value the generated control represents.

Pascal

```
property ParamName: string;
```

C++

```
__property UnicodeString ParamName;__property UnicodeString ParamName;
```

Description

Use ParamName to bind the Web data input component to a parameter in the XML data packet or supply an application-specific name to the value for this control. ParamName is only used when the IsQueryField method returns true. By default, the controls in a query form represent parameter values. However, they can be used to represent any named value.

If ParamName identifies one of the parameters of an XML broker, the XML broker property of the parent Web item identifies the broker that uses the parameter.

See Also

FieldName (see page 2094)

IsQueryField (see page 2081)

Params

XMLBroker (see page 2187)

Setting Web Item Properties

37.1.39.4.9 TQueryText.ReadOnly Property

Indicates whether end users can use the generated control to set values.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use ReadOnly to indicate whether users can change the value displayed by the generated control.

37.1.39.4.10 TQueryText.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.39.4.11 TQueryText.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.39.4.12 TQueryText.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.39.4.13 TQueryText.Text Property

Specifies the initial value of the generated control.

Pascal

```
property Text: string;
```

C++

```
__property UnicodeString Text;
```

Description

Use Text to initialize the value that appears in the generated display control.

See Also

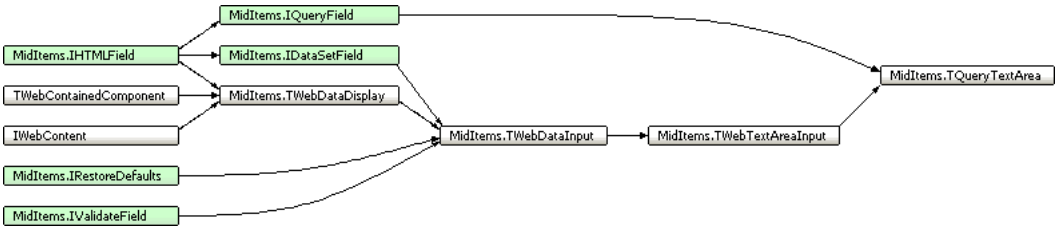
FieldName (see page 2094)

ParamName (see page 2094)

37.1.40 TQueryTextArea Class

TQueryTextArea creates a multi-line HTML input control to represent named parameter.

Class Hierarchy



Pascal

```
TQueryTextArea = class(TWebTextAreaInput, IQueryField);
```

C++

```
class TQueryTextArea : public TWebTextAreaInput, public IQueryField;
```

File

MidItems

Description

Add TQueryTextArea as the child of a TQueryFieldGroup (see page 1965) component to add a multi-line edit control that represents a parameter value. By default, the children of TQueryFieldGroup (see page 1965) represent the parameters of an XML broker. However, TQueryTextArea can represent any named value that the InternetExpress application uses.

Use the properties and methods of TQueryTextArea to

Specify the field or parameter whose value the control displays.

Assign an initial value to the control.

Specify a caption that describes the value of the control.

Customize the appearance of the control.

Fit the control into the tab order of the generated HTML form.

See Also

TQueryFieldGroup (see page 1965)

TXMLDisplayGroup (see page 2176)

TXMLBroker

37.1.40.1 TQueryTextArea Members

The following tables list the members exposed by TQueryTextArea.

Protected Methods

	Name	Description
	AddAttributes (see page 2014)	This is AddAttributes, a member of class TQueryTextArea.
	Content (see page 2076)	Returns the HTML that defines the display control.
	ControlContent (see page 2014)	Returns the HTML that defines the display control (without its caption).
	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
	EventContent (see page 2142)	This is EventContent, a member of class TWebTextAreaInput.
	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetText (see page 2014)	This is GetText, a member of class TQueryTextArea.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.

	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
	ImplGetRowSetFieldAttributes (see page 2143)	This is ImplGetRowSetFieldAttributes, a member of class TWebTextAreaInput.
	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
	RestoreFieldDefaults (see page 2143)	This is RestoreFieldDefaults, a member of class TWebTextAreaInput.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetLines (see page 2015)	This is SetLines, a member of class TQueryTextArea.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	SetText (see page 2015)	This is SetText, a member of class TQueryTextArea.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.

Public Methods

	Name	Description
	Create (see page 2013)	Creates and initializes an instance of TQueryTextArea (see page 2010).
	Destroy (see page 2013)	Frees an instance of TQueryTextArea (see page 2010).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetParamName (see page 2218)	This is GetParamName, a member of class IQueryField.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 2015)	Indicates whether the component generates a control that can appear in a query form.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	SetParamName (see page 2218)	This is SetParamName, a member of class IQueryField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	Text (see page 2219)	This is Text, a member of class IQueryField.

Published Properties

	Name	Description
	Caption (see page 2016)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2016)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 2017)	Specifies where the label appears relative to the generated display control.
	Custom (see page 2017)	Represents a tag option and its value in the tag for the display control.

	DisplayRows (see page 2017)	Specifies the height of the generated control.
	DisplayWidth (see page 2018)	Specifies the horizontal size of the generated control.
	FieldName (see page 2018)	Specifies the name of the field whose value the generated control represents.
	Lines (see page 2019)	Specifies the initial value that appears in the multi-line edit control.
	ParamName (see page 2019)	Specifies the name of the parameter whose value the generated control represents.
	ReadOnly (see page 2020)	Indicates whether end users can use the generated control to set values.
	Style (see page 2020)	Adds style attributes to the generated display control.
	StyleRule (see page 2020)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2021)	Determines the position of the generated control in the HTML form's tab order.
	Wrap (see page 2021)	Specifies how the generated control treats lines that are longer than the display width.

37.1.40.2 TQueryTextArea.Create Constructor

Creates and initializes an instance of TQueryTextArea (see page 2010).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TQueryTextArea(TComponent * AOwner);
```

Description

Call Create to instantiate TQueryTextArea (see page 2010) at runtime. TQueryTextArea (see page 2010) components added at design time using the Web Page editor are instantiated automatically.

37.1.40.3 TQueryTextArea.Destroy Destructor

Frees an instance of TQueryTextArea (see page 2010).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TQueryTextArea();
```

Description

Do not call Destroy directly in an application. Instead, use the Free method, which checks that the component reference is not nil, and only then calls Destroy.

See Also

System::TObject::Free

37.1.40.4 TQueryTextArea Methods

The methods of the TQueryTextArea class are listed here.

Protected Methods

	Name	Description
	AddAttributes (see page 2014)	This is AddAttributes, a member of class TQueryTextArea.
	ControlContent (see page 2014)	Returns the HTML that defines the display control (without its caption).

	GetText (see page 2014)	This is GetText, a member of class TQueryTextArea.
	SetLines (see page 2015)	This is SetLines, a member of class TQueryTextArea.
	SetText (see page 2015)	This is SetText, a member of class TQueryTextArea.

Public Methods

	Name	Description
	IsQueryField (see page 2015)	Indicates whether the component generates a control that can appear in a query form.

37.1.40.4.1 TQueryTextArea.AddAttributes Method

Pascal

```
procedure AddAttributes(var Attrs: string); override;
```

C++

```
virtual __fastcall AddAttributes(UnicodeString Attrs);
```

Description

This is AddAttributes, a member of class TQueryTextArea.

37.1.40.4.2 TQueryTextArea.ControlContent Method

Returns the HTML that defines the display control (without its caption).

Pascal

```
function ControlContent(Options: TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString ControlContent(TWebContentOptions Options);
```

Description

ControlContent is used internally to generate the HTML for the display control. This method does not generate any HTML for the control's label, which is handled separately by the FormatCaption method.

TWebDataDisplay introduces ControlContent as an abstract (pure virtual) method. Descendant classes override this method to generate the appropriate type of display component.

Options contains a set of flags that limit what can be included in the generated HTML.

ControlContent returns a string of HTML defining the display control.

See Also

ImplContent (see page 2080)

EventContent (see page 2077)

FormatCaption (see page 2077)

37.1.40.4.3 TQueryTextArea.GetText Method

Pascal

```
function GetText: string;
```

C++

```
__fastcall UnicodeString GetText();
```

Description

This is GetText, a member of class TQueryTextArea.

37.1.40.4.4 TQueryTextArea.IsQueryField Method

Indicates whether the component generates a control that can appear in a query form.

Pascal

```
class function IsQueryField: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsQueryField();
```

Description

Call IsQueryField to determine whether a Web display component generates a control that represents a named value in a query form (TQueryForm (see page 1971)). IsQueryField returns true, because TQueryTextArea (see page 2010) is designed specifically for TQueryFieldGroup (see page 1965) to use.

See Also

IsColumn (see page 2081)

TQueryFieldGroup (see page 1965)

TQueryForm (see page 1971)

37.1.40.4.5 TQueryTextArea.SetLines Method

Pascal

```
procedure SetLines(const Value: TStrings);
```

C++

```
__fastcall SetLines(const TStrings Value);
```

Description

This is SetLines, a member of class TQueryTextArea.

37.1.40.4.6 TQueryTextArea.SetText Method

Pascal

```
procedure SetText(const Value: string);
```

C++

```
__fastcall SetText(const UnicodeString Value);
```

Description

This is SetText, a member of class TQueryTextArea.

37.1.40.5 TQueryTextArea Properties

The properties of the TQueryTextArea class are listed here.

Published Properties

	Name	Description
	Caption (see page 2016)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2016)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 2017)	Specifies where the label appears relative to the generated display control.
	Custom (see page 2017)	Represents a tag option and its value in the tag for the display control.
	DisplayRows (see page 2017)	Specifies the height of the generated control.
	DisplayWidth (see page 2018)	Specifies the horizontal size of the generated control.
	FieldName (see page 2018)	Specifies the name of the field whose value the generated control represents.
	Lines (see page 2019)	Specifies the initial value that appears in the multi-line edit control.
	ParamName (see page 2019)	Specifies the name of the parameter whose value the generated control represents.
	ReadOnly (see page 2020)	Indicates whether end users can use the generated control to set values.
	Style (see page 2020)	Adds style attributes to the generated display control.
	StyleRule (see page 2020)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2021)	Determines the position of the generated control in the HTML form's tab order.
	Wrap (see page 2021)	Specifies how the generated control treats lines that are longer than the display width.

37.1.40.5.1 TQueryTextArea.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.40.5.2 TQueryTextArea.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.40.5.3 TQueryTextArea.CaptionPosition Property

Specifies where the label appears relative to the generated display control.

Pascal

```
property CaptionPosition: TCaptionPosition;
```

C++

```
__property TCaptionPosition CaptionPosition;
```

Description

Use CaptionPosition to indicate where the label should appear relative to the control that displays information.

Note: CaptionPosition has no effect on column controls. The position of column labels is always in the table header.

See Also

CaptionAttributes (see page 2084)

Caption (see page 2083)

37.1.40.5.4 TQueryTextArea.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.40.5.5 TQueryTextArea.DisplayRows Property

Specifies the height of the generated control.

Pascal

```
property DisplayRows: Integer;
```

C++

```
__property int DisplayRows;
```

Description

Use DisplayRows to indicate the desired height of the generated HTML control. DisplayRows specifies the number of lines of text that can be displayed in the control. This value is assigned to the ROWS option of the generated TEXTAREA tag.

See Also

DisplayWidth (see page 2144)

37.1.40.5.6 TQueryTextArea.DisplayWidth Property

Specifies the horizontal size of the generated control.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

Description

Use DisplayWidth to indicate the desired width of the generated HTML control. DisplayWidth specifies the number of characters that can be displayed in one line of the control. This value is assigned to the COLS option of the generated TEXTAREA tag.

See Also

DisplayRows (see page 2143)

37.1.40.5.7 TQueryTextArea.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.40.5.8 TQueryTextArea.Lines Property

Specifies the initial value that appears in the multi-line edit control.

Pascal

```
property Lines: TStrings;
```

C++

```
__property TStrings Lines;
```

Description

Use Lines to initialize the value that appears in the generated display control.

Each string in the TStrings object designates a line in the text value. This may affect the formatting, depending on the value of Wrap (see page 2021).

See Also

FieldName (see page 2094)

ParamName (see page 2094)

Wrap (see page 2144)

37.1.40.5.9 TQueryTextArea.ParamName Property

Specifies the name of the parameter whose value the generated control represents.

Pascal

```
property ParamName: string;
```

C++

```
__property UnicodeString ParamName;__property UnicodeString ParamName;
```

Description

Use ParamName to bind the Web data input component to a parameter in the XML data packet or supply an application-specific name to the value for this control. ParamName is only used when the IsQueryField method returns true. By default, the controls in a query form represent parameter values. However, they can be used to represent any named value.

If ParamName identifies one of the parameters of an XML broker, the XML broker property of the parent Web item identifies the broker that uses the parameter.

See Also

FieldName (see page 2094)

IsQueryField (see page 2081)

Params

XMLBroker (see page 2187)

Setting Web Item Properties

37.1.40.5.10 TQueryTextArea.ReadOnly Property

Indicates whether end users can use the generated control to set values.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use ReadOnly to indicate whether users can change the value displayed by the generated control. This value determines whether there is a READONLY option added to the generated TEXTAREA tag.

37.1.40.5.11 TQueryTextArea.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.40.5.12 TQueryTextArea.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.40.5.13 TQueryTextArea.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.40.5.14 TQueryTextArea.Wrap Property

Specifies how the generated control treats lines that are longer than the display width.

Pascal

```
property Wrap: TTextAreaWrap;
```

C++

```
__property TTextAreaWrap Wrap;
```

Description

Use Wrap to specify how the generated control formats lines that are longer than the number of characters specified by DisplayWidth. TWebTextAreaInput uses Wrap to add a WRAP option to the generated TEXTAREA tag.

Wrap can have any of the following values:

Value	WRAP option value	Meaning
wrOff	OFF	The control does not wrap long lines. Only newline characters that are part of the text string cause the lines in the control to wrap.
wrPhysical	PHYSICAL	The control formats its text value by wrapping long lines so that no text is hidden. The control inserts carriage returns into the text string.
wrVirtual	VIRTUAL	The control formats its text value by wrapping long lines so that no text is hidden but does not change the actual value of the text string.

See Also

DisplayWidth (see page 2144)

37.1.41 TResetQueryButton Class

TResetQueryButton generates an HTML control that backs out changes to the values on a query form.

Class Hierarchy

Pascal

```
TResetQueryButton = class(TQueryButton);
```

C++

```
class TResetQueryButton : public TQueryButton;
```

File

MidItems

Description

Add TResetQueryButton as the child of a TQueryButtons (see page 1962) component (or a layout group that is a child of a TQueryButtons (see page 1962) component) to enable the generated HTML form to undo edits made in the query form's display fields. The resulting HTML button's action backs out all edits to the values displayed on the query form.

Use the properties of TResetQueryButton to customize the appearance of the button.

See Also

TXMLBroker

TQueryForm (see page 1971)

TQueryButtons (see page 1962)

TLayoutGroup (see page 1939)

37.1.41.1 TResetQueryButton Members

The following tables list the members exposed by TResetQueryButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1955)	This is DefaultCaption, a member of class TQueryButton.
	InputType (see page 1956)	This is InputType, a member of class TQueryButton.

Protected Methods

	Name	Description
	Content (see page 2060)	Returns the HTML that defines the button control.
V	EventContent (see page 2024)	This is EventContent, a member of class TResetQueryButton.
V	GetCaption (see page 1956)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.

	GetHTMLControlName (see page 1957)	This is GetHTMLControlName, a member of class TQueryButton.
	GetHTMLForm (see page 1957)	This is GetHTMLForm, a member of class TQueryButton.
	GetInputType (see page 1957)	This is GetInputType, a member of class TQueryButton.
	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
	GetXMLComponent (see page 1957)	This is GetXMLComponent, a member of class TQueryButton.
	ImplContent (see page 1958)	Provides the underlying implementation for the Content method.
	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
	Notification (see page 1958)	Forwards notification messages to all owned components.
	ParentXMLComponent (see page 1959)	This is ParentXMLComponent, a member of class TQueryButton.
	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetXMLComponent (see page 1959)	This is SetXMLComponent, a member of class TQueryButton.
	SetXMLUseParent (see page 1959)	This is SetXMLUseParent, a member of class TQueryButton.

Public Methods

	Name	Description
	Create (see page 2023)	Creates and initializes an instance of TResetQueryButton (see page 2022).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.

Published Properties

	Name	Description
	Caption (see page 1959)	Specifies the caption that appears on the button.
	Custom (see page 1960)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1960)	Adds style attributes to the generated button control.
	StyleRule (see page 1961)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1961)	Associates the button with the query form whose display controls it influences.
	XMLUseParent (see page 1962)	Specifies whether the button is associated with its parent's query form.

37.1.41.2 TResetQueryButton.Create Constructor

Creates and initializes an instance of TResetQueryButton (see page 2022).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TResetQueryButton(TComponent * AOwner);
```

Description

Call Create to instantiate TResetQueryButton (see page 2022) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.41.3 TResetQueryButton Methods

The methods of the TResetQueryButton class are listed here.

Protected Methods

	Name	Description
	EventContent (see page 2024)	This is EventContent, a member of class TResetQueryButton.

37.1.41.3.1 TResetQueryButton.EventContent Method

Pascal

```
function EventContent(Options: TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString EventContent(TWebContentOptions Options);
```

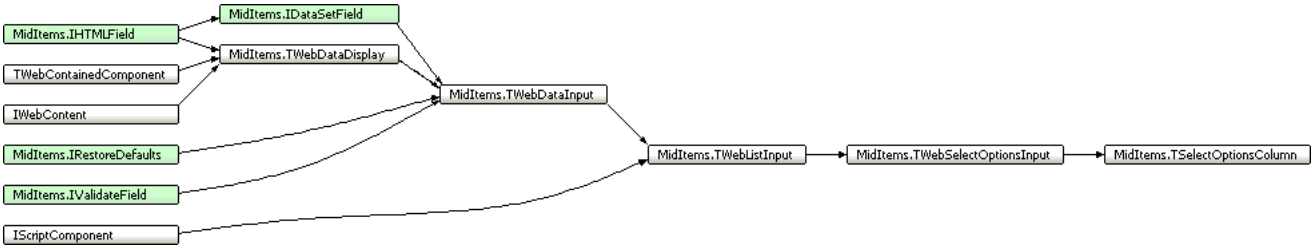
Description

This is EventContent, a member of class TResetQueryButton.

37.1.42 TSelectOptionsColumn Class

TSelectOptionsColumn creates an HTML list control to represent a field value in a data grid.

Class Hierarchy



Pascal

```
TSelectOptionsColumn = class(TWebSelectOptionsInput);
```

C++

```
class TSelectOptionsColumn : public TWebSelectOptionsInput;
```

File

MidItems

Description

Add TSelectOptionsColumn as the child of a TDataGrid (see page 1850) component to add a list control as the display control for a column. TSelectOptionsColumn represents the value of a field in the TDataGrid (see page 1850)'s associated dataset.

Use the properties and methods of TSelectOptionsColumn to

Specify the field whose value the list box displays.

Specify the possible values for the field from which the user can choose.

Specify the caption that appears at the top of the column.

Customize the appearance of the list box.

Fit the list box into the tab order of the generated HTML form.

See Also

TDataGrid (see page 1850)

TXMLDisplayGroup (see page 2176)

37.1.42.1 TSelectOptionsColumn Members

The following tables list the members exposed by TSelectOptionsColumn.

Protected Methods

	Name	Description
V	AddAttributes (see page 2131)	This is AddAttributes, a member of class TWebSelectOptionsInput.
	AddElements (see page 2117)	This is AddElements, a member of class TWebListInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
A	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
V	ControlContent (see page 2117)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
V	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
V	EventContent (see page 2132)	This is EventContent, a member of class TWebSelectOptionsInput.
V	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
A	FormatInputs (see page 2117)	This is FormatInputs, a member of class TWebListInput.
V	FormatInputs (see page 2132)	This is FormatInputs, a member of class TWebSelectOptionsInput.
V	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
V	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
V	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
V	GetSelectIndex (see page 2132)	This is GetSelectIndex, a member of class TWebSelectOptionsInput.
	GetSubComponents (see page 2118)	This is GetSubComponents, a member of class TWebListInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplAddElements (see page 2118)	This is ImplAddElements, a member of class TWebListInput.
V	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
V	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
V	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
V	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
V	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
V	Notification (see page 2118)	Forwards notification messages to all owned components.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
V	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
V	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
V	SetCaption (see page 2092)	Sets the value of the Caption property.

	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetDataSet (see page 2119)	This is SetDataSet, a member of class TWebListInput.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetItems (see page 2119)	This is SetItems, a member of class TWebListInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	SetValues (see page 2119)	This is SetValues, a member of class TWebListInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
V	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
V	Create (see page 2116)	Creates and initializes an instance of TWebListInput (see page 2113).
V	Create (see page 2131)	Creates and initializes an instance of TWebSelectOptionsInput (see page 2128).
V	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
V	Destroy (see page 2116)	Frees an instance of TWebListInput (see page 2113).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
V	IsColumn (see page 2027)	Indicates whether the component generates a control that can appear in the column of a data grid.
V	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

Published Properties

	Name	Description
	Caption (see page 2028)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2028)	Customizes the appearance of the generated control's label.
	Custom (see page 2028)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 2029)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayRows (see page 2029)	Specifies the number of rows that appear in the generated list box.
	FieldName (see page 2030)	Specifies the name of the field whose value the generated control represents.
	Items (see page 2030)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 2031)	Indicates which field in DataSet supplies the possible selections for the generated control.
	Style (see page 2031)	Adds style attributes to the generated display control.
	StyleRule (see page 2032)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2032)	Determines the position of the generated control in the HTML form's tab order.
	Values (see page 2032)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 2033)	Indicates which field in DataSet supplies the possible values for the generated control.

37.1.42.2 TSelectOptionsColumn Methods

The methods of the TSelectOptionsColumn class are listed here.

Public Methods

	Name	Description
	IsColumn (see page 2027)	Indicates whether the component generates a control that can appear in the column of a data grid.

37.1.42.2.1 TSelectOptionsColumn.IsColumn Method

Indicates whether the component generates a control that can appear in the column of a data grid.

Pascal

```
class function IsColumn: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsColumn();
```

Description

Call IsColumn to determine whether a Web display component generates a control that represents a column in a data grid (TDataGrid (see page 1850)). IsColumn returns true, because TSelectOptionsColumn (see page 2024) is designed specifically for TDataGrid (see page 1850) to use.

See Also

IsColumn (see page 2081)

TDataGrid (see page 1850)

37.1.42.3 TSelectOptionsColumn Properties

The properties of the TSelectOptionsColumn class are listed here.

Published Properties

	Name	Description
	Caption (see page 2028)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2028)	Customizes the appearance of the generated control's label.
	Custom (see page 2028)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 2029)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayRows (see page 2029)	Specifies the number of rows that appear in the generated list box.
	FieldName (see page 2030)	Specifies the name of the field whose value the generated control represents.
	Items (see page 2030)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 2031)	Indicates which field in DataSet supplies the possible selections for the generated control.
	Style (see page 2031)	Adds style attributes to the generated display control.
	StyleRule (see page 2032)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2032)	Determines the position of the generated control in the HTML form's tab order.
	Values (see page 2032)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 2033)	Indicates which field in DataSet supplies the possible values for the generated control.

37.1.42.3.1 TSelectOptionsColumn.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.42.3.2 TSelectOptionsColumn.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.42.3.3 TSelectOptionsColumn.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator

key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.42.3.4 TSelectOptionsColumn.DataSet Property

Specifies a dataset that supplies the possible values listed in the generated control.

Pascal

```
property DataSet: TDataSet;
```

C++

```
__property TDataSet DataSet;
```

Description

Use DataSet to specify a dataset from which the component can extract the selections that appear to the user and the underlying values they represent. The ItemsField property specifies the field in this dataset that lists the selections that appear in the generated control. The ValuesField property specifies the field in this dataset that lists the corresponding values that are assigned to the control.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

ItemsField (see page 2120)

ValuesField (see page 2121)

Items (see page 2120)

Values (see page 2121)

37.1.42.3.5 TSelectOptionsColumn.DisplayRows Property

Specifies the number of rows that appear in the generated list box.

Pascal

```
property DisplayRows: Integer;
```

C++

```
__property int DisplayRows;
```

Description

Use DisplayRows to specify how many rows appear in the generated list control. If DisplayRows is less than the number of items, the generated list displays only DisplayRows options at one time, but can scroll to display the hidden items. If DisplayRows is 1, the generated control becomes a drop-down list box.

See Also

ItemsField (see page 2120)

Items (see page 2120)

37.1.42.3.6 TSelectOptionsColumn.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.42.3.7 TSelectOptionsColumn.Items Property

Lists the possible selections that appear in the generated control.

Pascal

```
property Items: TStrings;
```

C++

```
__property TStrings Items;
```

Description

Use Items to supply a list of possible selections to the generated control. The end-user can choose from these selections to assign a value to the generated control. The value that is assigned comes from the corresponding entry in the Values property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items and Values properties, use the DataSet, ItemsField, and ValuesField properties.

See Also

ItemsField (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.42.3.8 TSelectOptionsColumn.ItemsField Property

Indicates which field in DataSet supplies the possible selections for the generated control.

Pascal

```
property ItemsField: string;
```

C++

```
__property UnicodeString ItemsField;
```

Description

Use ItemsField to specify which field in the dataset specified by the DataSet property contains selections that appear in the generated control. The end-user can choose from these selections to assign a value to the generated control. The value that is assigned comes from the field specified by the ValuesField property.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.42.3.9 TSelectOptionsColumn.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.42.3.10 TSelectOptionsColumn.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.42.3.11 TSelectOptionsColumn.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.42.3.12 TSelectOptionsColumn.Values Property

Lists the possible values that can be assigned to the control.

Pascal

```
property Values: TStrings;
```

C++

```
__property TStrings Values;
```

Description

Use Values to supply a list of possible values for the generated control. When the end-user chooses one of the selections listed by the Items property, the control is assigned the corresponding value in the Values property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items and Values properties, use the DataSet, ItemsField, and ValuesField properties.

See Also

Items (see page 2120)

ItemsField (see page 2120)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.42.3.13 TSelectOptionsColumn.ValuesField Property

Indicates which field in DataSet supplies the possible values for the generated control.

Pascal

```
property ValuesField: string;
```

C++

```
__property UnicodeString ValuesField;
```

Description

Use ValuesField to specify which field in the dataset specified by the DataSet property contains values that can be assigned to the generated control. When the end-user selects the corresponding value form the ItemsField field, the control gets the value in the ValuesField field.

Note: If the list of possible values is not available in a dataset, use the Items and Values properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

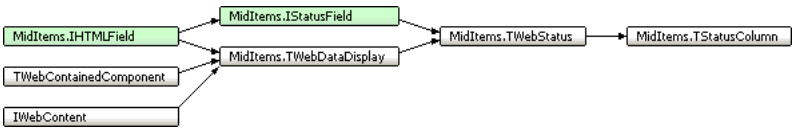
DataSet (see page 2119)

ItemsField (see page 2120)

37.1.43 TStatusColumn Class

TStatusColumn generates an HTML control that indicates whether a record in a data grid has been modified by the end user.

Class Hierarchy



Pascal

```
TStatusColumn = class(TWebStatus);
```

C++

```
class TStatusColumn : public TWebStatus;
```

File

MidItems

Description

Add TStatusColumn as the child of a TDataGrid (see page 1850) component to provide feedback about whether records in the grid have been modified. The generated control displays an M if the user has edited a record, an I if the user has inserted the record, and blank if the user has made no changes to the record.

Use the properties and methods of TStatusColumn to

Specify the caption that appears at the top of the column.

Customize the appearance of the control.

Fit the control into the tab order of the generated HTML form.

See Also

TDataGrid (see page 1850)

TXMLDisplayGroup (see page 2176)

37.1.43.1 TStatusColumn Members

The following tables list the members exposed by TStatusColumn.

Protected Methods

	Name	Description
	Content (see page 2076)	Returns the HTML that defines the display control.
A	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
V	ControlContent (see page 2135)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
V	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
V	GetCaption (see page 2136)	Returns the value of the Caption property.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
V	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetHTMLControlName (see page 2136)	Returns the name that other components can use to refer to the generated display control.
V	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
V	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.

	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.

Public Methods

	Name	Description
V	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
V	Create (see page 2135)	Creates and initializes an instance of TWebStatus (see page 2133).
V	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	Identifier (see page 2136)	Returns the name under which record status information is stored in the XML data packet.
V	IsColumn (see page 2035)	Indicates whether the component generates a control that can appear in the column of a data grid.
V	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.

Public Properties

	Name	Description
	CaptionPosition (see page 2138)	Specifies where the label appears relative to the generated display control.
	DisplayWidth (see page 2138)	Specifies the width, in characters, of the generated HTML control.
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	TabIndex (see page 2086)	Determines the position of the generated control in the HTML form's tab order.

Published Properties

	Name	Description
	Caption (see page 2036)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2036)	Customizes the appearance of the generated control's label.
	Custom (see page 2037)	Represents a tag option and its value in the tag for the display control.
	Style (see page 2037)	Adds style attributes to the generated display control.
	StyleRule (see page 2038)	Applies a style in the page producer's style sheet to the generated display control.

37.1.43.2 TStatusColumn Methods

The methods of the TStatusColumn class are listed here.

Public Methods

	Name	Description
V	IsColumn (see page 2035)	Indicates whether the component generates a control that can appear in the column of a data grid.

37.1.43.2.1 TStatusColumn.IsColumn Method

Indicates whether the component generates a control that can appear in the column of a data grid.

Pascal

```
class function IsColumn: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsColumn();
```

Description

Call IsColumn to determine whether a Web item generates a control that represents a column in a data grid (TDataGrid (see page 1850)). IsColumn returns true, because TStatusColumn (see page 2033) is designed specifically for TDataGrid (see page 1850) to use.

See Also

IsColumn (see page 2081)

TDataGrid (see page 1850)

37.1.43.3 TStatusColumn Properties

The properties of the TStatusColumn class are listed here.

Published Properties

	Name	Description
	Caption (see page 2036)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2036)	Customizes the appearance of the generated control's label.
	Custom (see page 2037)	Represents a tag option and its value in the tag for the display control.
	Style (see page 2037)	Adds style attributes to the generated display control.
	StyleRule (see page 2038)	Applies a style in the page producer's style sheet to the generated display control.

37.1.43.3.1 TStatusColumn.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.43.3.2 TStatusColumn.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are

added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.43.3.3 TStatusColumn.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.43.3.4 TStatusColumn.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.43.3.5 TStatusColumn.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.44 TSubmitQueryButton Class

TSubmitQueryButton generates an HTML control that backs out changes to the values on a query form.

Class Hierarchy

Pascal

```
TSubmitQueryButton = class(TQueryButton);
```

C++

```
class TSubmitQueryButton : public TQueryButton;
```

File

MidItems

Description

Add TSubmitQueryButton as the child of a TQueryButtons (see page 1962) component (or a layout group that is a child of a TQueryButtons (see page 1962) component) to enable the generated HTML form to send the values entered in the query form back to the Web application. When the user presses the generated button, the values in the query form are sent to the Web application, using the path info specified by the query form's Action property.

Use the properties of TSubmitQueryButton to customize the appearance of the button.

See Also

TXMLBroker

TQueryForm (see page 1971)

TQueryButtons (see page 1962)

TLayoutGroup (see page 1939)

TCustomQueryForm (see page 1841)

37.1.44.1 TSubmitQueryButton Members

The following tables list the members exposed by TSubmitQueryButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1955)	This is DefaultCaption, a member of class TQueryButton.
	InputType (see page 1956)	This is InputType, a member of class TQueryButton.

Protected Methods

	Name	Description
	Content (see page 2060)	Returns the HTML that defines the button control.
V	EventContent (see page 2040)	This is EventContent, a member of class TSubmitQueryButton.
V	GetCaption (see page 1956)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
V	GetHTMLControlName (see page 1957)	This is GetHTMLControlName, a member of class TQueryButton.
	GetHTMLForm (see page 1957)	This is GetHTMLForm, a member of class TQueryButton.
V	GetInputType (see page 1957)	This is GetInputType, a member of class TQueryButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
	GetXMLComponent (see page 1957)	This is GetXMLComponent, a member of class TQueryButton.
V	ImplContent (see page 1958)	Provides the underlying implementation for the Content method.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 1958)	Forwards notification messages to all owned components.
	ParentXMLComponent (see page 1959)	This is ParentXMLComponent, a member of class TQueryButton.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetXMLComponent (see page 1959)	This is SetXMLComponent, a member of class TQueryButton.
	SetXMLUseParent (see page 1959)	This is SetXMLUseParent, a member of class TQueryButton.

Public Methods

	Name	Description
V	Create (see page 2040)	Creates and initializes an instance of TSubmitQueryButton (see page 2038).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.

Published Properties

	Name	Description
	Caption (see page 1959)	Specifies the caption that appears on the button.
	Custom (see page 1960)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1960)	Adds style attributes to the generated button control.
	StyleRule (see page 1961)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1961)	Associates the button with the query form whose display controls it influences.
	XMLUseParent (see page 1962)	Specifies whether the button is associated with its parent's query form.

37.1.44.2 TSubmitQueryButton.Create Constructor

Creates and initializes an instance of TSubmitQueryButton (see page 2038).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TSubmitQueryButton(TComponent * AOwner);
```

Description

Call Create to instantiate TSubmitQueryButton (see page 2038) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.44.3 TSubmitQueryButton Methods

The methods of the TSubmitQueryButton class are listed here.

Protected Methods

	Name	Description
	EventContent (see page 2040)	This is EventContent, a member of class TSubmitQueryButton.

37.1.44.3.1 TSubmitQueryButton.EventContent Method

Pascal

```
function EventContent(Options: TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString EventContent(TWebContentOptions Options);
```

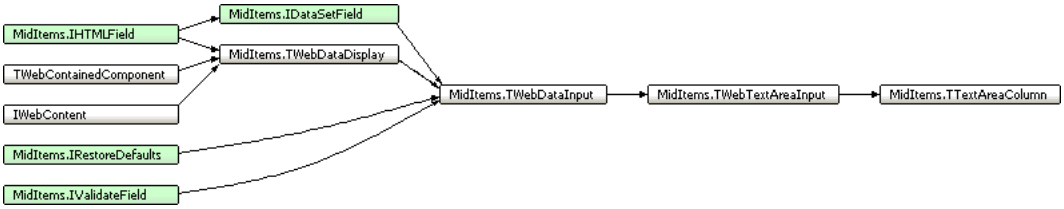
Description

This is EventContent, a member of class TSubmitQueryButton.

37.1.45 TTextAreaColumn Class

TTextAreaColumn creates a multi-line HTML input control to represent a field value in a data grid.

Class Hierarchy



Pascal

```
TTextAreaColumn = class(TWebTextAreaInput);
```

C++

```
class TTextAreaColumn : public TWebTextAreaInput;
```

File

MidItems

Description

Add TTextAreaColumn as the child of a TDataGrid (see page 1850) component to add a multi-line edit control as the display control for a column. TTextAreaColumn represents the value of a field in the TDataGrid (see page 1850)'s associated dataset.

Use the properties and methods of TTextAreaColumn to

Specify the field whose value the control displays.

Specify the caption that appears at the top of the column.

Customize the appearance of the control.

Fit the control into the tab order of the generated HTML form.

See Also

IWebContent

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

TQueryTextArea (see page 2010)

37.1.45.1 TTextAreaColumn Members

The following tables list the members exposed by TTextAreaColumn.

Protected Methods

	Name	Description
	AddAttributes (see page 2142)	This is AddAttributes, a member of class TWebTextAreaInput.

	Content (see page 2076)	Returns the HTML that defines the display control.
A	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
V	ControlContent (see page 2142)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
V	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
V	EventContent (see page 2142)	This is EventContent, a member of class TWebTextAreaInput.
V	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
V	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
V	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
V	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
V	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
V	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
V	ImplGetRowSetFieldAttributes (see page 2143)	This is ImplGetRowSetFieldAttributes, a member of class TWebTextAreaInput.
V	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
V	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
V	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
V	RestoreFieldDefaults (see page 2143)	This is RestoreFieldDefaults, a member of class TWebTextAreaInput.
V	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
V	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
V	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
V	Create (see page 2141)	Creates and initializes an instance of TWebTextAreaInput (see page 2138).
V	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.

	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
V	IsColumn (see page 2043)	Indicates whether the component generates a control that can appear in the column of a data grid.
V	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

Published Properties

	Name	Description
	Caption (see page 2044)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2044)	Customizes the appearance of the generated control's label.
	Custom (see page 2045)	Represents a tag option and its value in the tag for the display control.
	DisplayRows (see page 2045)	Specifies the height of the generated control.
	DisplayWidth (see page 2046)	Specifies the horizontal size of the generated control.
	FieldName (see page 2046)	Specifies the name of the field whose value the generated control represents.
	ReadOnly (see page 2046)	Indicates whether end users can use the generated control to set values.
	Style (see page 2047)	Adds style attributes to the generated display control.
	StyleRule (see page 2047)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2048)	Determines the position of the generated control in the HTML form's tab order.
	Wrap (see page 2048)	Specifies how the generated control treats lines that are longer than the display width.

37.1.45.2 TTextAreaColumn Methods

The methods of the TTextAreaColumn class are listed here.

Public Methods

	Name	Description
V	IsColumn (see page 2043)	Indicates whether the component generates a control that can appear in the column of a data grid.

37.1.45.2.1 TTextAreaColumn.IsColumn Method

Indicates whether the component generates a control that can appear in the column of a data grid.

Pascal

```
class function IsColumn: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsColumn();
```

Description

Call IsColumn to determine whether a Web display component generates a control that represents a column in a data grid (TDataGrid (see page 1850)). IsColumn returns true, because TTextAreaColumn (see page 2041) is designed specifically for TDataGrid (see page 1850) to use.

See Also

IsColumn (see page 2081)

TDataGrid (see page 1850)

37.1.45.3 TTextAreaColumn Properties

The properties of the TTextAreaColumn class are listed here.

Published Properties

	Name	Description
	Caption (see page 2044)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2044)	Customizes the appearance of the generated control's label.
	Custom (see page 2045)	Represents a tag option and its value in the tag for the display control.
	DisplayRows (see page 2045)	Specifies the height of the generated control.
	DisplayWidth (see page 2046)	Specifies the horizontal size of the generated control.
	FieldName (see page 2046)	Specifies the name of the field whose value the generated control represents.
	ReadOnly (see page 2046)	Indicates whether end users can use the generated control to set values.
	Style (see page 2047)	Adds style attributes to the generated display control.
	StyleRule (see page 2047)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2048)	Determines the position of the generated control in the HTML form's tab order.
	Wrap (see page 2048)	Specifies how the generated control treats lines that are longer than the display width.

37.1.45.3.1 TTextAreaColumn.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.45.3.2 TTextAreaColumn.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.45.3.3 TTextAreaColumn.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.45.3.4 TTextAreaColumn.DisplayRows Property

Specifies the height of the generated control.

Pascal

```
property DisplayRows: Integer;
```

C++

```
__property int DisplayRows;
```

Description

Use DisplayRows to indicate the desired height of the generated HTML control. DisplayRows specifies the number of lines of text that can be displayed in the control. This value is assigned to the ROWS option of the generated TEXTAREA tag.

See Also

DisplayWidth (see page 2144)

37.1.45.3.5 TTextAreaColumn.DisplayWidth Property

Specifies the horizontal size of the generated control.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

Description

Use DisplayWidth to indicate the desired width of the generated HTML control. DisplayWidth specifies the number of characters that can be displayed in one line of the control. This value is assigned to the COLS option of the generated TEXTAREA tag.

See Also

DisplayRows (see page 2143)

37.1.45.3.6 TTextAreaColumn.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.45.3.7 TTextAreaColumn.ReadOnly Property

Indicates whether end users can use the generated control to set values.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use ReadOnly to indicate whether users can change the value displayed by the generated control. This value determines whether there is a READONLY option added to the generated TEXTAREA tag.

37.1.45.3.8 TTextAreaColumn.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.45.3.9 TTextAreaColumn.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.45.3.10 TTextAreaColumn.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.45.3.11 TTextAreaColumn.Wrap Property

Specifies how the generated control treats lines that are longer than the display width.

Pascal

```
property Wrap: TTextAreaWrap;
```

C++

```
__property TTextAreaWrap Wrap;
```

Description

Use Wrap to specify how the generated control formats lines that are longer than the number of characters specified by DisplayWidth. TWebTextAreaInput uses Wrap to add a WRAP option to the generated TEXTAREA tag.

Wrap can have any of the following values:

Value	WRAP option value	Meaning
wrOff	OFF	The control does not wrap long lines. Only newline characters that are part of the text string cause the lines in the control to wrap.
wrPhysical	PHYSICAL	The control formats its text value by wrapping long lines so that no text is hidden. The control inserts carriage returns into the text string.
wrVirtual	VIRTUAL	The control formats its text value by wrapping long lines so that no text is hidden but does not change the actual value of the text string.

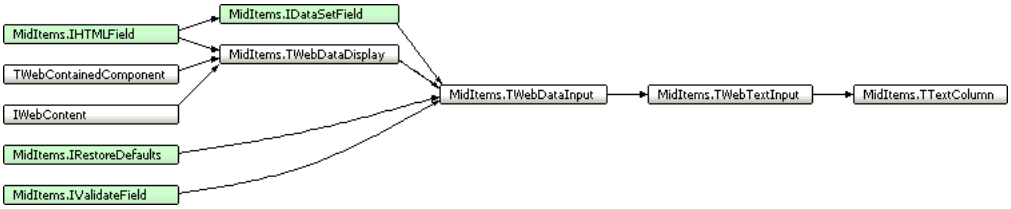
See Also

DisplayWidth (see page 2144)

37.1.46 TTextColumn Class

TTextColumn creates a single-line HTML input control to represent a field value in a data grid.

Class Hierarchy



Pascal

```
TTextColumn = class(TWebTextInput);
```

C++

```
class TTextColumn : public TWebTextInput;
```

File

MidItems

Description

Add TTextColumn as the child of a TDataGrid (see page 1850) component to add an edit control as the display control for a column. TTextColumn represents the value of a field in the TDataGrid (see page 1850)'s associated dataset.

Use the properties and methods of TTextColumn to

Specify the field whose value the control displays.

Specify the caption that appears at the top of the column.

Customize the appearance of the edit control.

Fit the control into the tab order of the generated HTML form.

See Also

TDataGrid (see page 1850)

TXMLDisplayGroup (see page 2176)

37.1.46.1 TTextColumn Members

The following tables list the members exposed by TTextColumn.

Protected Methods

	Name	Description
	AddAttributes (see page 2148)	This is AddAttributes, a member of class TWebTextInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
	ControlContent (see page 2148)	Returns the HTML that defines the display control (without its caption).
	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.

	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
	ImplGetRowSetFieldAttributes (see page 2149)	This is ImplGetRowSetFieldAttributes, a member of class TWebTextInput.
	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
	RestoreFieldDefaults (see page 2149)	This is RestoreFieldDefaults, a member of class TWebTextInput.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
	Create (see page 2147)	Creates and initializes an instance of TWebTextInput (see page 2145).
	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	IsColumn (see page 2051)	Indicates whether the component generates a control that can appear in the column of a data grid.
	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.

	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

Published Properties

	Name	Description
	Caption (see page 2052)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2052)	Customizes the appearance of the generated control's label.
	Custom (see page 2053)	Represents a tag option and its value in the tag for the display control.
	DisplayWidth (see page 2053)	Specifies the horizontal size of the generated control.
	FieldName (see page 2053)	Specifies the name of the field whose value the generated control represents.
	ReadOnly (see page 2054)	Indicates whether end users can use the generated control to set values.
	Style (see page 2054)	Adds style attributes to the generated display control.
	StyleRule (see page 2055)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2055)	Determines the position of the generated control in the HTML form's tab order.

37.1.46.2 TTextColumn Methods

The methods of the TTextColumn class are listed here.

Public Methods

	Name	Description
V	IsColumn (see page 2051)	Indicates whether the component generates a control that can appear in the column of a data grid.

37.1.46.2.1 TTextColumn.IsColumn Method

Indicates whether the component generates a control that can appear in the column of a data grid.

Pascal

```
class function IsColumn: Boolean; override;
```

C++

```
virtual __fastcall Boolean IsColumn();
```

Description

Call IsColumn to determine whether a Web display component generates a control that represents a column in a data grid (TDataGrid (see page 1850)). IsColumn returns true, because TTextColumn (see page 2049) is designed specifically for TDataGrid (see page 1850) to use.

See Also

IsColumn (see page 2081)

TDataGrid (see page 1850)

37.1.46.3 TTextColumn Properties

The properties of the TTextColumn class are listed here.

Published Properties

	Name	Description
	Caption (see page 2052)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2052)	Customizes the appearance of the generated control's label.
	Custom (see page 2053)	Represents a tag option and its value in the tag for the display control.
	DisplayWidth (see page 2053)	Specifies the horizontal size of the generated control.
	FieldName (see page 2053)	Specifies the name of the field whose value the generated control represents.
	ReadOnly (see page 2054)	Indicates whether end users can use the generated control to set values.
	Style (see page 2054)	Adds style attributes to the generated display control.
	StyleRule (see page 2055)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2055)	Determines the position of the generated control in the HTML form's tab order.

37.1.46.3.1 TTextColumn.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.46.3.2 TTextColumn.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.46.3.3 TTextColumn.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.46.3.4 TTextColumn.DisplayWidth Property

Specifies the horizontal size of the generated control.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

Description

Use DisplayWidth to indicate the desired width of the generated HTML control. DisplayWidth specifies the number of characters that can be displayed in the control.

Note: DisplayWidth controls the size of the generated control, not the size of the text that users can enter. If the user enters a string longer than DisplayWidth characters, the text scrolls in the control.

37.1.46.3.5 TTextColumn.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.46.3.6 TTextColumn.ReadOnly Property

Indicates whether end users can use the generated control to set values.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use ReadOnly to indicate whether users can change the value displayed by the generated control.

37.1.46.3.7 TTextColumn.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.46.3.8 TTextColumn.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.46.3.9 TTextColumn.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.47 TUndoButton Class

TUndoButton generates an HTML control that backs out changes to the current record.

Class Hierarchy

Pascal

```
TUndoButton = class(TDataSetButton);
```

C++

```
class TUndoButton : public TDataSetButton;
```

File

MidItems

Description

Add TUndoButton as the child of a TDataNavigator (see page 1855) component (or a layout group that is a child of a TDataNavigator (see page 1855) component) to enable the generated HTML form to undo edits made in an associated display control or controls. The resulting HTML button's action backs out all edits to the current record.

Use the properties of TUndoButton to customize the appearance of the button and associate it with a Web item that generates the display controls.

See Also

TXMLBroker

TDataNavigator (see page 1855)

TLayoutGroup (see page 1939)

37.1.47.1 TUndoButton Members

The following tables list the members exposed by TUndoButton.

Protected Fields

	Name	Description
	DefaultCaption (see page 1862)	This is DefaultCaption, a member of class TDataSetButton.
	XMLMethodName (see page 1862)	This is XMLMethodName, a member of class TDataSetButton.

Protected Methods

	Name	Description
	AddElements (see page 1862)	This is AddElements, a member of class TDataSetButton.
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 1862)	Returns the value of the Caption property.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
V	GetMethodName (see page 1863)	This is GetMethodName, a member of class TDataSetButton.
	GetSubComponents (see page 1863)	This is GetSubComponents, a member of class TDataSetButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
V	ImplAddElements (see page 1863)	This is ImplAddElements, a member of class TDataSetButton.

	ImplContent (see page 1863)	Provides the underlying implementation for the Content method.
	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
	Notification (see page 2194)	Forwards notification messages to all owned components.
	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
	Create (see page 2057)	Creates and initializes an instance of TUndoButton (see page 2055).
	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

Published Properties

	Name	Description
	Caption (see page 1864)	Specifies the caption that appears on the button.
	Custom (see page 1865)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 1865)	Adds style attributes to the generated button control.
	StyleRule (see page 1865)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 1866)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 1866)	Specifies whether the button is associated with its parent's display object.

37.1.47.2 TUndoButton.Create Constructor

Creates and initializes an instance of TUndoButton (see page 2055).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TUndoButton(TComponent * AOwner);
```

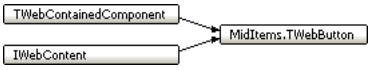
Description

Call Create to instantiate TUndoButton (see page 2055) at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.48 TWebButton Class

TWebButton is the base class for Web items that create HTML button controls.

Class Hierarchy



Pascal

```
TWebButton = class(TWebContainedComponent, IWebContent);
```

C++

```
class TWebButton : public TWebContainedComponent, public IWebContent;
```

File

MidItems

Description

Use TWebButton as a base class for components that generate the HTML statements defining a specialized button control in a WebSnap or InternetExpress application. TWebButton introduces the properties and methods for

Generating the HTML code to define the button control.

Defining a button caption.

Specifying tag attributes of the HTML button control, including those defined by a style sheet.

In order to allow other Web items to use the HTML generated by the button control, TWebButton implements the IWebContent interface.

Do not create instances of TWebButton. TWebButton relies on an abstract or, in C++ terminology, pure virtual method to generate the HTML statements defining a button control. Instead, create descendants of TWebButton that represent buttons with specific purposes.

See Also

TDeleteButton (see page 1867)

TFirstButton (see page 1913)

TInsertButton (see page 1935)

TLastButton (see page 1937)

TNextButton (see page 1943)

TNextPageButton (see page 1945)

TPostButton (see page 1947)

TPriorButton (see page 1949)

TPriorPageButton (see page 1951)

TUndoButton (see page 2055)

TResetQueryButton (see page 2022)

TSubmitQueryButton (see page 2038)

TApplyUpdatesButton (see page 1798)

IWebContent

37.1.48.1 TWebButton Members

The following tables list the members exposed by TWebButton.

Protected Methods

	Name	Description
	Content (see page 2060)	Returns the HTML that defines the button control.
	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.

Public Methods

	Name	Description
	Create (see page 2059)	Creates and initializes an instance of TWebButton (see page 2058).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.

37.1.48.2 TWebButton.Create Constructor

Creates and initializes an instance of TWebButton (see page 2058).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TWebButton(TComponent * AOwner);
```

Description

Call Create to instantiate a Web button at runtime. Web buttons added at design time using the Web Page editor are instantiated automatically.

37.1.48.3 TWebButton Methods

The methods of the TWebButton class are listed here.

Protected Methods

	Name	Description
	Content (see page 2060)	Returns the HTML that defines the button control.
	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.

	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.

37.1.48.3.1 TWebButton.Content Method

Returns the HTML that defines the button control.

Pascal

```
function Content(Options: TWebContentOptions; Layout: TLayout): string;
```

C++

```
__fastcall UnicodeString Content(TWebContentOptions Options, TLayout Layout);
```

Description

Content implements the IWebContent method of the same name. This method allows the Web button's parent to extract the HTML defining the button.

Content calls the virtual ImplContent (see page 2061) method to provide the content it returns. Each TWebButton (see page 2058) descendant overrides ImplContent (see page 2061) to generate the definition of its particular button control.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the Web button should format the HTML control it generates. It implements the ILayoutWebContent interface.

Content returns a string of HTML defining the button control.

See Also

ImplContent (see page 2061)

IWebContent

ILayoutWebContent

37.1.48.3.2 TWebButton.GetCaption Method

Returns the value of the Caption (see page 2062) property.

Pascal

```
function GetCaption: string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetCaption();
```

Description

GetCaption is the protected read implementation for the Caption (see page 2062) property.

See Also

Caption (see page 2062)

SetCaption (see page 2061)

37.1.48.3.3 TWebButton.GetLayoutAttributes Method

Returns the layout object that specifies how the generated control should look.

Pascal

```
function GetLayoutAttributes: TLayoutAttributes; virtual;
```

C++

```
virtual __fastcall TLayoutAttributes GetLayoutAttributes();
```

Description

GetLayoutAttributes is used internally by Web items to allow the layout object supplied by a parent (as a parameter to the Content (see page 2060) method) to take into account layout properties of the child component.

As implemented by TWebButton (see page 2058), GetLayoutAttributes returns nil (Delphi) or NULL (C++), because the layout of buttons does not make use of layout attributes.

See Also

Content (see page 2060)

37.1.48.3.4 TWebButton.ImplContent Method

Provides the underlying implementation for the Content (see page 2060) method.

Pascal

```
function ImplContent(Options: TWebContentOptions; Layout: TLayout): string; virtual; abstract;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout Layout) = 0;
```

Description

The Content (see page 2060) method calls ImplContent to generate the HTML that defines the button control. TWebButton (see page 2058) introduces ImplContent as an abstract or, in C++ terminology, pure virtual method. TWebButton (see page 2058) descendants must override ImplContent to generate the definition of their particular button control.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the Web button should format the HTML control it generates. It implements the ILayoutWebContent interface.

ImplContent returns a string of HTML defining the button control.

See Also

Content (see page 2060)

ILayoutWebContent

37.1.48.3.5 TWebButton.SetCaption Method

Sets the value of the Caption (see page 2062) property.

Pascal

```
procedure SetCaption(const AValue: string); virtual;
```

C++

```
virtual __fastcall SetCaption(const UnicodeString AValue);
```

Description

SetCaption is the protected write implementation for the Caption (see page 2062) property.

See Also

Caption (see page 2062)

GetCaption (see page 2060)

37.1.48.4 TWebButton Properties

The properties of the TWebButton class are listed here.

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.

37.1.48.4.1 TWebButton.Caption Property

Specifies the caption that appears on the button.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the caption that appears on the generated HTML button. The value of Caption is used as the value of a VALUE parameter on the HTML INPUT tag generated to implement the button.

See Also

Custom (see page 2062)

GetCaption (see page 2060)

SetCaption (see page 2061)

37.1.48.4.2 TWebButton.Custom Property

Represents a tag option and its value in the generated INPUT tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

Use Custom to represent one or more options that qualify the INPUT tag generated by the Web button. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to add an ALIGN option that aligns the caption

and a TABINDEX tag to control tab order, set Custom to

```
ALIGN=TOP TABINDEX=2
```

See Also

Caption (see page 2062)

Style (see page 2063)

37.1.48.4.3 TWebButton.Style Property

Adds style attributes to the generated button control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to add a STYLE option that qualifies the INPUT tag generated by the Web button. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated INPUT tag as the value of a STYLE option.

See Also

Custom (see page 2062)

StyleRule (see page 2063)

37.1.48.4.4 TWebButton.StyleRule Property

Applies a style in the page producer's style sheet to the button.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated button control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

Note: To apply a style definition that is not included in the style sheet, use the button's Style (see page 2063) property.

See Also

Style (see page 2063)

Styles (see page 2272)

StylesFile (see page 2273)

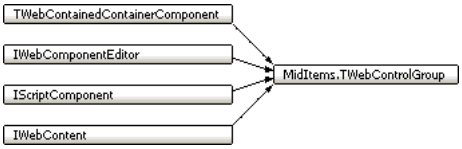
Styles

StylesFile

37.1.49 TWebControlGroup Class

TWebControlGroup is the base class for Web items that arrange HTML controls.

Class Hierarchy



Pascal

```
TWebControlGroup = class(TWebContainedContainerComponent, IWebComponentEditor,
IScriptComponent, IWebContent);
```

C++

```
class TWebControlGroup : public TWebContainedContainerComponent, public IWebComponentEditor,
public IScriptComponent, public IWebContent;
```

File

MidItems

Description

Use TWebControlGroup as a base class for components that generate organized portions of the HTML document containing one or more related HTML controls. TWebControlGroup introduces the properties and methods for managing a collection of Web items that generate the HTML controls that make up the group.

In order to let other Web items to use the HTML that defines the organized collection of controls, TWebControlGroup implements the IWebContent interface.

Do not create instances of TWebControlGroup. TWebControlGroup manages the Web items that generate HTML defining controls, but does not combine the generated content into any content of its own. Instead, use a descendant of TWebControlGroup for a set of specialized controls. These descendants are listed in the See also list.

See Also

TDataNavigator (see page 1855)

TQueryButtons (see page 1962)

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

IWebContent

TAdapterFieldGroup

TAdapterGrid

TAdapterCommandColumn

TAdapterForm

37.1.49.1 TWebControlGroup Members

The following tables list the members exposed by TWebControlGroup.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
V	ImplAddElements (see page 2069)	Provides the underlying implementation for the AddElements (see page 2066) method.
V	ImplCanAddClass (see page 2070)	Provides the underlying implementation for the CanAddClass (see page 2067) method.
V	ImplContent (see page 2070)	Provides the underlying implementation for the Content (see page 2068) method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.

Public Methods

	Name	Description
V	Create (see page 2065)	Creates and initializes an instance of TWebControlGroup (see page 2064).
V	Destroy (see page 2066)	Frees an instance of TWebControlGroup (see page 2064).

Public Properties

	Name	Description
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.

37.1.49.2 TWebControlGroup.Create Constructor

Creates and initializes an instance of TWebControlGroup (see page 2064).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TWebControlGroup(TComponent * AOwner);
```

Description

Call Create to instantiate TWebControlGroup (see page 2064) at runtime. TWebControlGroup (see page 2064) descendants added at design time are instantiated automatically.

37.1.49.3 TWebControlGroup.Destroy Destructor

Frees an instance of TWebControlGroup (see page 2064).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TWebControlGroup();
```

Description

Destroy removes the Web control group from its parent's list of components that generate HTML before freeing its own resources.

37.1.49.4 TWebControlGroup Methods

The methods of the TWebControlGroup class are listed here.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
V	ImplAddElements (see page 2069)	Provides the underlying implementation for the AddElements (see page 2066) method.
V	ImplCanAddClass (see page 2070)	Provides the underlying implementation for the CanAddClass (see page 2067) method.
V	ImplContent (see page 2070)	Provides the underlying implementation for the Content (see page 2068) method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.

37.1.49.4.1 TWebControlGroup.AddElements Method

Used internally to generate any script declarations needed by the group of controls.

Pascal

```
procedure AddElements(AddIntf: IAddScriptElements);
```

C++

```
__fastcall AddElements(IAddScriptElements AddIntf);
```

Description

The page producer calls AddElements when it generates the block of declarations for the script language it uses. This method allows the generated group of controls to use classes and methods in the scripting language of the page producer.

AddElements calls on the virtual ImplAddElements (see page 2069) method, which uses the provided interface to return its

declarations. Each TWebControlGroup (see page 2064) descendant overrides ImplAddElements (see page 2069) to add any declarations it needs.

See Also

ImplAddElements (see page 2069)

TAdapterPageProducer

TInetXPageProducer (see page 2285)

37.1.49.4.2 TWebControlGroup.CanAddClass Method

Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.

Pascal

```
function CanAddClass(AParent: TComponent; AClass: TClass): Boolean;
```

C++

```
__fastcall Boolean CanAddClass(TComponent * AParent, TClass AClass);
```

Description

The Web page editor calls CanAddClass to determine what Web items can be added to supply content for the TWebControlGroup (see page 2064) descendant. CanAddClass calls the virtual ImplCanAddClass (see page 2070) method to determine whether a class of Web item is valid.

AParent is the TWebControlGroup (see page 2064) descendant to whose WebFieldControls (see page 2072) property the Web item will be added.

AClass is the class of the proposed child Web item.

CanAddClass returns true if the TWebControlGroup (see page 2064) descendant can use content generated by a component of type AClass, false otherwise.

See Also

ImplCanAddClass (see page 2070)

WebFieldControls (see page 2072)

37.1.49.4.3 TWebControlGroup.ChangeFieldControls Method

Responds when the group of HTML controls changes.

Pascal

```
procedure ChangeFieldControls(Sender: TObject);
```

C++

```
__fastcall ChangeFieldControls(TObject * Sender);
```

Description

ChangeFieldControls is called automatically when the list of Web items that generate HTML controls in this group changes. (It is assigned as the OnChange event handler to the TWebComponentList object that implements the WebFieldControls (see page 2072) property.)

ChangeFieldControls sets the DefaultWebComponents (see page 2072) property to nil (Delphi) or NULL (C++) when an application explicitly assigns child items to the WebFieldControls (see page 2072) property.

See Also

OnChange

WebFieldControls (see page 2072)

DefaultWebComponents (see page 2072)

37.1.49.4.4 TWebControlGroup.Content Method

Returns the HTML that defines the group of HTML controls.

Pascal

```
function Content(Options: TWebContentOptions; ParentLayout: TLayout): string;
```

C++

```
__fastcall UnicodeString Content(TWebContentOptions Options, TLayout ParentLayout);
```

Description

Content implements the IWebContent method of the same name. This method allows the TWebControlGroup (see page 2064) object's parent to extract the HTML that defines and lays out a set of controls.

Content calls the virtual ImplContent (see page 2070) method to provide the HTML it returns. Each TWebControlGroup (see page 2064) descendant overrides ImplContent (see page 2070) to generate its particular group of controls. The generated HTML can make use of any script declarations supplied by the AddElements (see page 2066) method.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the component should format the HTML it generates. It implements the ILayoutWebContent interface.

Content returns a string of HTML defining the group of controls.

See Also

ImplContent (see page 2070)

IWebContent

AddElements (see page 2066)

ILayoutWebContent

37.1.49.4.5 TWebControlGroup.GetDefaultWebComponents Method

Returns the value of the DefaultWebComponents (see page 2072) property.

Pascal

```
function GetDefaultWebComponents: TWebComponentList; override;
```

C++

```
virtual __fastcall TWebComponentList GetDefaultWebComponents();
```

Description

GetDefaultWebComponents returns the value of DefaultWebComponents (see page 2072). When no Web items have explicitly been added to the control group, this is a list of controls that appear by default. Once an application explicitly assigns controls to the WebFieldControls (see page 2072) property, GetDefaultWebComponents returns nil (Delphi) or NULL (C++).

See Also

WebFieldControls (see page 2072)

DefaultWebComponents (see page 2072)

SetDefaultWebComponents (see page 2071)

37.1.49.4.6 TWebControlGroup.GetLayoutAttributes Method

Returns an object that specifies how the generated group of controls should look.

Pascal

```
function GetLayoutAttributes: TLayoutAttributes; virtual;
```

C++

```
virtual __fastcall TLayoutAttributes GetLayoutAttributes();
```

Description

GetLayoutAttributes is used internally by Web items to allow the layout object supplied by a parent (as a parameter to the Content (see page 2068) method) to take into account layout properties of the child component.

As implemented by TWebControlGroup (see page 2064), GetLayoutAttributes returns nil (Delphi) or NULL (C++), because TWebControlGroup (see page 2064) descendants not make use of layout attributes.

See Also

Content (see page 2068)

37.1.49.4.7 TWebControlGroup.GetSubComponents Method

Returns the value of the WebFieldControls (see page 2072) property.

Pascal

```
function GetSubComponents: TObject;
```

C++

```
__fastcall TObject * GetSubComponents();
```

Description

GetSubComponents returns the TWebControlGroup (see page 2064)'s list of Web items that it uses to generate HTML content. This method is part of the mechanism that fits TWebControlGroup (see page 2064) into a hierarchy of Web items, each of which generates part of a larger HTML document.

See Also

WebFieldControls (see page 2072)

ImplGetSubComponents (see page 2071)

37.1.49.4.8 TWebControlGroup.ImplAddElements Method

Provides the underlying implementation for the AddElements (see page 2066) method.

Pascal

```
procedure ImplAddElements(AddIntf: IAddScriptElements); virtual;
```

C++

```
virtual __fastcall ImplAddElements(IAddScriptElements AddIntf);
```

Description

The AddElements (see page 2066) method calls ImplAddElements to add any script declarations by calling the AddIntf interface. These declarations allow the generated display control to call script methods in response to user events by ensuring that the required methods are declared.

As implemented in TWebControlGroup (see page 2064), ImplAddElements does nothing. TWebControlGroup (see page 2064) descendants override ImplAddElements to add any required declarations.

See Also

AddElements (see page 2066)

37.1.49.4.9 TWebControlGroup.ImplCanAddClass Method

Provides the underlying implementation for the CanAddClass (see page 2067) method.

Pascal

```
function ImplCanAddClass(AParent: TComponent; AClass: TClass): Boolean; virtual;
```

C++

```
virtual __fastcall Boolean ImplCanAddClass(TComponent * AParent, TClass AClass);
```

Description

The CanAddClass (see page 2067) method calls ImpleCanAddClass to determine whether the TWebControlGroup (see page 2064) descendant can add a component of type AClass to its WebComponents property list. AParent is the TWebControlGroup (see page 2064) instance to which the component will be added.

In TWebControlGroup (see page 2064), ImplCanAddClass always returns true. TWebControlGroup (see page 2064) descendants override ImplCanAddClass to indicate what types of Web Items they can use when generating their HTML content.

See Also

CanAddClass (see page 2067)

WebFieldControls (see page 2072)

37.1.49.4.10 TWebControlGroup.ImplContent Method

Provides the underlying implementation for the Content (see page 2068) method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; virtual;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content (see page 2068) method calls ImplContent to generate the group of HTML controls. As implemented in TWebControlGroup (see page 2064), ImplContent returns an empty string. TWebControlGroup (see page 2064) descendants override it to generate HTML for the group of controls.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the TWebControlGroup (see page 2064)

should format the HTML controls it generates. It implements the ILayoutWebContent interface.

ImplContent returns the generated HTML.

See Also

Content (see page 2068)

AddElements (see page 2066)

ILayoutWebContent

37.1.49.4.11 TWebControlGroup.ImplGetSubComponents Method

Provides the underlying implementation for the GetSubComponents (see page 2069) method.

Pascal

```
function ImplGetSubComponents: TObject; virtual;
```

C++

```
virtual __fastcall TObject * ImplGetSubComponents();
```

Description

The GetSubComponents (see page 2069) method calls ImplGetSubComponents to retrieve the value of WebFieldControls (see page 2072) as a TObject. TWebControlGroup (see page 2064) descendants override this method if the list is not simply the value of WebFieldControls (see page 2072). For example, some TWebControlGroup (see page 2064) descendants supply a default value (DefaultWebComponents (see page 2072)) when the WebFieldControls (see page 2072) property list is empty.

ImplGetSubComponents returns the TWebComponentList object that contains all subitems of the Web control group.

See Also

GetSubComponents (see page 2069)

DefaultWebComponents (see page 2072)

WebFieldControls (see page 2072)

37.1.49.4.12 TWebControlGroup.SetDefaultWebComponents Method

Sets the value of the DefaultWebComponents (see page 2072) property.

Pascal

```
procedure SetDefaultWebComponents(Value: TWebComponentList); override;
```

C++

```
virtual __fastcall SetDefaultWebComponents(TWebComponentList Value);
```

Description

SetDefaultWebComponents assigns a list of Web items as the value of the DefaultWebComponents (see page 2072) property. This list specifies the controls that appear in the control group if the application does not explicitly assign any controls to the WebFieldControls (see page 2072) property.

See Also

DefaultWebComponents (see page 2072)

GetDefaultWebComponents (see page 2068)

WebFieldControls (see page 2072)

37.1.49.5 TWebControlGroup Properties

The properties of the TWebControlGroup class are listed here.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.

Public Properties

	Name	Description
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.

37.1.49.5.1 TWebControlGroup.DefaultWebComponents Property

Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.

Pascal

```
property DefaultWebComponents: TWebComponentList;
```

C++

```
__property TWebComponentList DefaultWebComponents;
```

Description

Most TWebControlGroup (see page 2064) descendants have a default set of values for the WebFieldControls (see page 2072) property.

Note: For example, on Windows, TDataNavigator (see page 1855) has a default set of buttons and TFieldGroup (see page 1869) defaults to a control for each field in the associated data packet.

TWebControlGroup (see page 2064) descendants that supply default sets of Web items store these default sets as the value of the protected DefaultWebComponents property. Then, if the value of WebFieldControls (see page 2072) is not set, the TWebControlGroup (see page 2064) descendant uses DefaultWebComponents instead.

See Also

WebFieldControls (see page 2072)

37.1.49.5.2 TWebControlGroup.WebFieldControls Property

Lists the components that generate HTML controls for the Web control group.

Pascal

```
property WebFieldControls: TWebComponentList;
```

C++

```
__property TWebComponentList WebFieldControls;
```

Description

WebFieldControls lists the components that generate the related HTML controls that the Web control group assembles into a single unit. Each of these components supports the IWebContent interface, and some can contain their own list of Web items that help them generate their content.

The type of items that can be added to the WebFieldControls list are restricted by the CanAddClass (see page 2067) method. TWebControlGroup (see page 2064) does not restrict the items in the list at all, but descendant classes add restrictions to limit the list to related types of controls.

Note: For example, on Windows, TDataNavigator (see page 1855) limits the list to button generators that are descendants of TDataSetButton (see page 1859) or TXMLButton (see page 2151).

At design time, add components to the WebFieldControls property by bringing up the Web page editor for the containing page producer. Select the Web control group in the editor, and use the editor to add child controls.

See Also

TInetXPageProducer (see page 2285)

TAdapterPageProducer

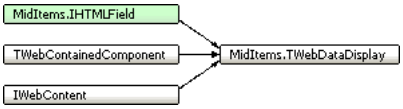
IWebContent

CanAddClass (see page 2067)

37.1.50 TWebDataDisplay Class

TWebDataDisplay is the base class for Web items that create HTML display controls.

Class Hierarchy



Pascal

```
TWebDataDisplay = class(TWebContainedComponent, IWebContent, IHTMLField);
```

C++

```
class TWebDataDisplay : public TWebContainedComponent, public IWebContent, public IHTMLField;
```

File

MidItems

Description

Use TWebDataDisplay as a base class for components that generate HTML controls which display information in a WebSnap or InternetExpress application. TWebDataDisplay descendants generate the individual controls that are organized by TAdapterGrid or TAdapterFieldGroup.

Note: On Windows, additional descendants, TDataGrid (see page 1850), TFieldGroup (see page 1869), or TQueryFieldGroup (see page 1965), can be used to generate the individual controls.

TWebDataDisplay introduces the properties and methods for

Generating a caption that describes the information displayed.

Customizing the appearance of the control that displays information.

Fitting the control into the tab order of the generated HTML form.

Fitting the component into a hierarchy of Web items that collectively generate the content of an HTML document.

In order to let other Web items to use the HTML control it generates, TWebDataDisplay implements the IWebContent interface.

Do not create instances of TWebDataDisplay. TWebDataDisplay relies on an abstract (pure virtual) method for generating the display control. Instead, use a descendant of TWebDataDisplay that generates an HTML control of the desired type.

See Also

IWebContent

TAdapterGrid

TAdapterFieldGroup

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

37.1.50.1 TWebDataDisplay Members

The following tables list the members exposed by TWebDataDisplay.

Protected Methods

	Name	Description
	Content (see page 2076)	Returns the HTML that defines the display control.
A	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
V	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
V	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
V	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
V	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.

Public Methods

	Name	Description
V	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).
V	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
V	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
V	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.

Public Properties

	Name	Description
	Caption (see page 2083)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2084)	Customizes the appearance of the generated control's label.
	Custom (see page 2084)	Represents a tag option and its value in the tag for the display control.
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	Style (see page 2085)	Adds style attributes to the generated display control.
	StyleRule (see page 2086)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2086)	Determines the position of the generated control in the HTML form's tab order.

37.1.50.2 TWebDataDisplay.Create Constructor

Creates and initializes an instance of TWebDataDisplay (see page 2073).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TWebDataDisplay(TComponent * AOwner);
```

Description

Call Create to instantiate TWebDataDisplay (see page 2073) at runtime. TWebDataDisplay (see page 2073) descendants added at design time are instantiated automatically.

37.1.50.3 TWebDataDisplay.Destroy Destructor

Frees an instance of TWebDataDisplay (see page 2073).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TWebDataDisplay();
```

Description

Do not call Destroy directly in an application. Instead, use the Free method, which checks that the component is not nil, and only then calls Destroy.

See Also

System::TObject::Free

37.1.50.4 TWebDataDisplay Methods

The methods of the TWebDataDisplay class are listed here.

Protected Methods

	Name	Description
	Content (see page 2076)	Returns the HTML that defines the display control.
A	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.

	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.

Public Methods

	Name	Description
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.

37.1.50.4.1 TWebDataDisplay.Content Method

Returns the HTML that defines the display control.

Pascal

```
function Content(Options: TWebContentOptions; ParentLayout: TLayout): string;
```

C++

```
__fastcall UnicodeString Content(TWebContentOptions Options, TLayout ParentLayout);
```

Description

Content implements the IWebContent method of the same name. This method allows the TWebDataDisplay (see page 2073) component's parent to extract the HTML defining the display control.

Content calls the virtual ImplContent (see page 2080) method to provide the HTML content that it returns. Each TWebDataDisplay (see page 2073) descendant overrides ImplContent (see page 2080) to generate the definition of its particular control.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the component should format the HTML control it generates. It supports the ILayoutWebContent interface.

Content returns a string of HTML defining the display control.

See Also

ImplContent (see page 2080)

IWebContent

ILayoutWebContent

37.1.50.4.2 TWebDataDisplay.ControlContent Method

Returns the HTML that defines the display control (without its caption).

Pascal

```
function ControlContent(Options: TWebContentOptions): string; virtual; abstract;
```

C++

```
virtual __fastcall UnicodeString ControlContent(TWebContentOptions Options) = 0;
```

Description

ControlContent is used internally to generate the HTML for the display control. This method does not generate any HTML for the control's label, which is handled separately by the FormatCaption (see page 2077) method.

TWebDataDisplay (see page 2073) introduces ControlContent as an abstract (pure virtual) method. Descendant classes override this method to generate the appropriate type of display component.

Options contains a set of flags that limit what can be included in the generated HTML.

ControlContent returns a string of HTML defining the display control.

See Also

ImplContent (see page 2080)

EventContent (see page 2077)

FormatCaption (see page 2077)

37.1.50.4.3 TWebDataDisplay.EventContent Method

Pascal

```
function EventContent(Options: TWebContentOptions): string; virtual;
```

C++

```
virtual __fastcall UnicodeString EventContent(TWebContentOptions Options);
```

Description

This is EventContent, a member of class TWebDataDisplay.

37.1.50.4.4 TWebDataDisplay.FormatCaption Method

Returns the HTML for the display control's caption.

Pascal

```
function FormatCaption: string; virtual;
```

C++

```
virtual __fastcall UnicodeString FormatCaption();
```

Description

FormatCaption generates the HTML for the display control's caption, using the attributes specified by the CaptionAttributes (see page 2084) property. FormatCaption returns the resulting string of HTML.

In addition, FormatCaption encodes the layout properties of the caption and the value of the CaptionPosition (see page 2084) property into the LayoutAttributes (see page 2085) property. The TWebDataDisplay (see page 2073) component uses this

value later to lay out the control and its caption appropriately.

See Also

Caption (see page 2083)

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

LayoutAttributes (see page 2085)

37.1.50.4.5 TWebDataDisplay.GetCaption Method

Returns the value of the Caption (see page 2083) property.

Pascal

```
function GetCaption: string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetCaption();
```

Description

GetCaption is the protected read implementation for the Caption (see page 2083) property.

See Also

Caption (see page 2083)

37.1.50.4.6 TWebDataDisplay.GetHTMLControlName Method

Returns the name that other components can use to refer to the generated display control.

Pascal

```
function GetHTMLControlName: string;
```

C++

```
__fastcall UnicodeString GetHTMLControlName();
```

Description

GetHTMLControlName returns the name of the generated control. This value is assigned to a NAME option on the generated HTML tag.

GetHTMLControlName returns the value supplied by the ImplGetHTMLControlName (see page 2081) method. Unlike GetHTMLControlName, ImplGetHTMLControlName (see page 2081) is a virtual method that descendants can override to supply a name in class-specific ways.

See Also

ImplGetHTMLControlName (see page 2081)

37.1.50.4.7 TWebDataDisplay.GetHTMLForm Method

Used internally to retrieve information about the form in which the display control is defined.

Pascal

```
function GetHTMLForm: IHTMLForm;
```

C++

```
__fastcall IHTMLForm GetHTMLForm();
```

Description

TWebDataDisplay (see page 2073) and its descendants use GetHTMLForm to obtain information about the form in which the generated display control appears. This information is used to generate code for calling form methods in response to user events.

GetHTMLForm walks up the hierarchy of Web items (using the GetParentComponent method) until it finds a component that can supply an interface for the HTML form.

See Also

GetParentComponent

37.1.50.4.8 TWebDataDisplay.GetLayoutAttributes Method

Returns the value of the LayoutAttributes (see page 2085) property.

Pascal

```
function GetLayoutAttributes: TLayoutAttributes; virtual;
```

C++

```
virtual __fastcall TLayoutAttributes GetLayoutAttributes();
```

Description

GetLayoutAttributes is used internally by Web items to allow the layout object supplied by a parent (as a parameter to the Content (see page 2076) method) to take into account layout properties of the child component.

GetLayoutAttributes returns the value of the LayoutAttributes (see page 2085) property.

See Also

Content (see page 2076)

LayoutAttributes (see page 2085)

37.1.50.4.9 TWebDataDisplay.GetXmlDisplayName Method

Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.

Pascal

```
function GetXmlDisplayName: string;
```

C++

```
__fastcall UnicodeString GetXmlDisplayName();
```

Description

GetXmlDisplayName is not available for cross-platform programming. It is introduced in TWebDataDisplay (see page 2073) for descendant classes that are only accessible on Windows.

On Windows, GetXmlDisplayName returns the name of the javascript object that groups the set of display controls which includes this TWebDataDisplay (see page 2073) descendant's generated control. The javascript object is generated by a TDataGrid (see page 1850) or TFieldGroup (see page 1869) component, and provides the display component's link to its XML data packet. Typically, the TDataGrid (see page 1850) or TFieldGroup (see page 1869) component is the parent of the TWebDataDisplay (see page 2073) component, but it may be higher in the parent hierarchy if the application uses layout

groups.

See Also

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

XMLBroker (see page 2187)

37.1.50.4.10 TWebDataDisplay.GetXmlRowSetName Method

Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.

Pascal

```
function GetXmlRowSetName: string;
```

C++

```
__fastcall UnicodeString GetXmlRowSetName();
```

Description

GetXmlRowSetName is not available for cross-platform programming. It is introduced in TWebDataDisplay (see page 2073) for descendant classes that are only accessible on Windows.

On Windows, GetXmlRowSetName is used when generating HTML that accesses data from the XML data packet. It returns the name of the rowset that represents the data in the Web data display.

As implemented in TWebDataDisplay (see page 2073), GetXmlRowSetName walks up the parent hierarchy until it finds a Web item associated with a rowset, and then returns the name of that rowset.

See Also

GetXmlDisplayName (see page 2079)

37.1.50.4.11 TWebDataDisplay.ImplContent Method

Provides the underlying implementation for the Content (see page 2076) method.

Pascal

```
function ImplContent(Options: TWebContentOptions; ParentLayout: TLayout): string; virtual;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout ParentLayout);
```

Description

The Content (see page 2076) method calls ImplContent to generate the HTML that defines the display control and its label. ImplContent formats the caption and combines this HTML with the results of the ControlContent (see page 2077) method. ControlContent (see page 2077) is an abstract (pure virtual) method in TWebDataDisplay (see page 2073). TWebDataDisplay (see page 2073) descendants override it to generate the definition of their particular display control.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the TWebDataDisplay (see page 2073) should format the HTML control it generates. It implements the ILayoutWebContent interface.

ImplContent returns a string of HTML defining the display control.

See Also

Content (see page 2076)

Caption (see page 2083)

ILayoutWebContent

37.1.50.4.12 TWebDataDisplay.ImplGetHTMLControlName Method

Returns the name that other components can use to refer to the generated display control.

Pascal

```
function ImplGetHTMLControlName: string; virtual;
```

C++

```
virtual __fastcall UnicodeString ImplGetHTMLControlName();
```

Description

ImplGetHTMLControlName returns the name of the generated control. This value is assigned to a NAME option on the generated HTML tag.

ImplGetHTMLControlName returns the value of the Name property. TWebDataDisplay (see page 2073) descendants can override this method to return a more meaningful value such as the name of the field or parameter displayed in the control.

See Also

Name

37.1.50.4.13 TWebDataDisplay.IsColumn Method

Indicates whether the component generates the controls that appear in the column of a grid.

Pascal

```
class function IsColumn: Boolean; virtual;
```

C++

```
virtual __fastcall Boolean IsColumn();
```

Description

Call IsColumn to determine whether a TWebDataDisplay (see page 2073) descendant generates the controls that appear in a single column of an adapter grid (TAdapterGrid) or data grid (TDataGrid (see page 1850)). As implemented in TWebDataDisplay (see page 2073), IsColumn returns false. Descendant classes that generate controls for the columns of a grid override this method to return true.

See Also

IsQueryField (see page 2081)

TDataGrid (see page 1850)

TAdapterGrid

37.1.50.4.14 TWebDataDisplay.IsQueryField Method

Indicates whether the component generates the controls that appear in a query form. Not used on Linux.

Pascal

```
class function IsQueryField: Boolean; virtual;
```

C++

```
virtual __fastcall Boolean IsQueryField();
```

Description

IsQueryField is not available for cross-platform programming. It is introduced in TWebDataDisplay (see page 2073) for descendant classes that are only used on Windows.

On Windows, Call IsQueryField to determine whether a TWebDataDisplay (see page 2073) descendant generates the controls that represent a named value in a query form (TQueryForm (see page 1971)). As implemented in TWebDataDisplay (see page 2073), IsQueryField returns false. Descendant classes that generate controls for a TQueryFieldGroup (see page 1965) component override this method to return true.

See Also

IsColumn (see page 2081)

TQueryFieldGroup (see page 1965)

TQueryForm (see page 1971)

37.1.50.4.15 TWebDataDisplay.LabelContent Method

Returns the HTML for the display control's caption.

Pascal

```
function LabelContent: string; virtual;
```

C++

```
virtual __fastcall UnicodeString LabelContent();
```

Description

The ImplContent (see page 2080) method calls LabelContent to provide the HTML for the label of the generated display control. LabelContent calls the FormatCaption (see page 2077) method to produce this HTML and returns the resulting string.

See Also

FormatCaption (see page 2077)

Caption (see page 2083)

ImplContent (see page 2080)

37.1.50.4.16 TWebDataDisplay.SetCaption Method

Sets the value of the Caption (see page 2083) property.

Pascal

```
procedure SetCaption(Value: string); virtual;
```

C++

```
virtual __fastcall SetCaption(UnicodeString Value);
```

Description

SetCaption is the protected write implementation for the Caption (see page 2083) property.

See Also

Caption (see page 2083)

37.1.50.4.17 TWebDataDisplay.SetCaptionAttributes Method

Sets the value of the CaptionAttributes (see page 2084) property.

Pascal

```
procedure SetCaptionAttributes(const Value: TCaptionAttributes);
```

C++

```
__fastcall SetCaptionAttributes(const TCaptionAttributes Value);
```

Description

SetCaptionAttributes is the protected write implementation for the CaptionAttributes (see page 2084) property.

See Also

CaptionAttributes (see page 2084)

37.1.50.5 TWebDataDisplay Properties

The properties of the TWebDataDisplay class are listed here.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.

Public Properties

	Name	Description
	Caption (see page 2083)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2084)	Customizes the appearance of the generated control's label.
	Custom (see page 2084)	Represents a tag option and its value in the tag for the display control.
	Style (see page 2085)	Adds style attributes to the generated display control.
	StyleRule (see page 2086)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2086)	Determines the position of the generated control in the HTML form's tab order.

37.1.50.5.1 TWebDataDisplay.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.50.5.2 TWebDataDisplay.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption (see page 2083) property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.50.5.3 TWebDataDisplay.CaptionPosition Property

Specifies where the label appears relative to the generated display control.

Pascal

```
property CaptionPosition: TCaptionPosition;
```

C++

```
__property TCaptionPosition CaptionPosition;
```

Description

Use CaptionPosition to indicate where the label should appear relative to the control that displays information.

Note: CaptionPosition has no effect on column controls. The position of column labels is always in the table header.

See Also

CaptionAttributes (see page 2084)

Caption (see page 2083)

37.1.50.5.4 TWebDataDisplay.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay (see page 2073) descendants generate different HTML tags to generate the control that displays

information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes (see page 2084) property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.50.5.5 TWebDataDisplay.LayoutAttributes Property

Encodes attributes that describe how the generated control should look.

Pascal

```
property LayoutAttributes: TLayoutAttributes;
```

C++

```
__property TLayoutAttributes LayoutAttributes;
```

Description

LayoutAttributes is used internally by Web items to allow the layout object supplied by a parent (as a parameter to the Content (see page 2076) method) to take into account layout properties of the child component.

LayoutAttributes encodes the information in the CaptionPosition (see page 2084) and CaptionAttributes (see page 2084) properties. Some descendant classes add information from the Style (see page 2085) and StyleRule (see page 2086) properties as well.

See Also

GetLayoutAttributes (see page 2079)

CaptionPosition (see page 2084)

Content (see page 2076)

Style (see page 2085)

CaptionAttributes (see page 2084)

StyleRule (see page 2086)

37.1.50.5.6 TWebDataDisplay.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes (see page 2084) property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.50.5.7 TWebDataDisplay.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style (see page 2085) property instead.

Note: To apply a style to the control's label, use the CaptionAttributes (see page 2084) property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.50.5.8 TWebDataDisplay.TabIndex Property

Determines the position of the generated control in the HTML form's tab order.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

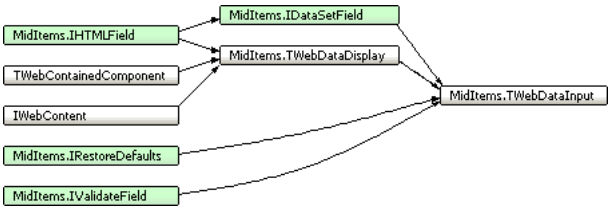
Description

Use TabIndex to control the order that focus moves between controls when the end-user presses the Tab key. TabIndex specifies the position of the control in the tab order. If TabIndex is less than 0, the HTML tag's TABINDEX option is not set.

37.1.51 TWebDataInput Class

TWebDataInput is the base class for Web items that create HTML display controls for entering information.

Class Hierarchy



Pascal

```
TWebDataInput = class(TWebDataDisplay, IRestoreDefaults, IDataSetField, IValidateField);
```

C++

```
class TWebDataInput : public TWebDataDisplay, public IRestoreDefaults, public IDataSetField,
public IValidateField;
```

File

MidItems

Description

Use TWebDataInput as a base class for components that generate HTML controls which display information in an InternetExpress application and into which users can enter values. TWebDataInput descendants generate controls that are organized by TDataGrid (see page 1850), TFieldGroup (see page 1869), or TQueryFieldGroup (see page 1965).

TWebDataInput defines or inherits properties and methods for

Generating a caption that describes the information displayed.

Customizing the appearance of the control that displays information.

Specifying a field or parameter whose value the control displays.

Fitting the control into the tab order of the generated HTML form.

Fitting the component into a hierarchy of Web items that collectively generate the content of an HTML document.

In order to let other Web items to use the HTML control it generates, TWebDataInput inherits the IWebContent interface from TWebDataDisplay (see page 2073).

Do not create instances of TWebDataInput. TWebDataInput relies on an abstract (pure virtual) method for generating the display control. Instead, use a descendant of TWebDataInput that generates an HTML control of the desired type.

See Also

IWebContent

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

37.1.51.1 TWebDataInput Members

The following tables list the members exposed by TWebDataInput.

Protected Methods

	Name	Description
	Content (see page 2076)	Returns the HTML that defines the display control.
A	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
V	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
V	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
V	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
V	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
V	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
V	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
V	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
V	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
V	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
V	Create (see page 2075)	Creates and initializes an instance of TWebDataDisplay (see page 2073).

	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.

Public Properties

	Name	Description
	Caption (see page 2083)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2084)	Customizes the appearance of the generated control's label.
	Custom (see page 2084)	Represents a tag option and its value in the tag for the display control.
	FieldName (see page 2094)	Specifies the name of the field whose value the generated control represents.
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	Style (see page 2085)	Adds style attributes to the generated display control.
	StyleRule (see page 2086)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2086)	Determines the position of the generated control in the HTML form's tab order.

37.1.51.2 TWebDataInput Methods

The methods of the TWebDataInput class are listed here.

Protected Methods

	Name	Description
	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

37.1.51.2.1 TWebDataInput.EventContent Method

Pascal

```
function EventContent(Options: TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString EventContent(TWebContentOptions Options);
```

Description

This is EventContent, a member of class TWebDataInput.

37.1.51.2.2 TWebDataInput.FindAssociatedField Method

Pascal

```
function FindAssociatedField(DataSet: IInetXWebDataSet): TField; virtual;
```

C++

```
virtual __fastcall TField FindAssociatedField(IInetXWebDataSet DataSet);
```

Description

This is FindAssociatedField, a member of class TWebDataInput.

37.1.51.2.3 TWebDataInput.GetCaption Method

Returns the value of the Caption property.

Pascal

```
function GetCaption: string; override;
```

C++

```
virtual __fastcall UnicodeString GetCaption();
```

Description

GetCaption is the protected read implementation for the Caption property.

See Also

Caption (see page 2083)

37.1.51.2.4 TWebDataInput.GetFieldName Method

Pascal

```
function GetFieldName: string;
```

C++

```
__fastcall UnicodeString GetFieldName();
```

Description

This is GetFieldName, a member of class TWebDataInput.

37.1.51.2.5 TWebDataInput.GetParamName Method

Pascal

```
function GetParamName: string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetParamName();
```

Description

This is GetParamName, a member of class TWebDataInput.

37.1.51.2.6 TWebDataInput.GetRowSetFieldAttributes Method

Pascal

```
function GetRowSetFieldAttributes(const FieldVarName: string): string;
```

C++

```
__fastcall UnicodeString GetRowSetFieldAttributes(const UnicodeString FieldVarName);
```

Description

This is GetRowSetFieldAttributes, a member of class TWebDataInput.

37.1.51.2.7 TWebDataInput.ImplGetFieldName Method

Pascal

```
function ImplGetFieldName: string; virtual;
```

C++

```
virtual __fastcall UnicodeString ImplGetFieldName();
```

Description

This is ImplGetFieldName, a member of class TWebDataInput.

37.1.51.2.8 TWebDataInput.ImplGetHTMLControlName Method

Returns the name that other components can use to refer to the generated display control.

Pascal

```
function ImplGetHTMLControlName: string; override;
```

C++

```
virtual __fastcall UnicodeString ImplGetHTMLControlName();
```

Description

ImplGetHTMLControlName returns the name of the generated control. This value is assigned to a NAME option on the generated HTML tag.

ImplGetHTMLControlName returns the value of the Name property. TWebDataDisplay descendants can override this method to return a more meaningful value such as the name of the field or parameter displayed in the control.

See Also

Name

37.1.51.2.9 TWebDataInput.ImplGetRowSetFieldAttributes Method

Pascal

```
function ImplGetRowSetFieldAttributes(const FieldVarName: string): string; virtual;
```

C++

```
virtual __fastcall UnicodeString ImplGetRowSetFieldAttributes(const UnicodeString
FieldVarName);
```

Description

This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.

37.1.51.2.10 TWebDataInput.ImplRestoreDefaults Method

Pascal

```
procedure ImplRestoreDefaults; virtual;
```

C++

```
virtual __fastcall ImplRestoreDefaults();
```

Description

This is ImplRestoreDefaults, a member of class TWebDataInput.

37.1.51.2.11 TWebDataInput.RestoreDefaults Method

Returns the Web item to a default state.

Pascal

```
procedure RestoreDefaults;
```

C++

```
__fastcall RestoreDefaults();
```

Description

The Web page editor calls RestoreDefaults when the user chooses the Restore Defaults command. Web items implement this method to return property values to a default state.

37.1.51.2.12 TWebDataInput.RestoreFieldDefaults Method

Pascal

```
procedure RestoreFieldDefaults(AField: TField); virtual;
```

C++

```
virtual __fastcall RestoreFieldDefaults(TField AField);
```

Description

This is RestoreFieldDefaults, a member of class TWebDataInput.

37.1.51.2.13 TWebDataInput.SetCaption Method

Sets the value of the Caption property.

Pascal

```
procedure SetCaption(Value: string); override;
```

C++

```
virtual __fastcall SetCaption(UnicodeString Value);
```

Description

SetCaption is the protected write implementation for the Caption property.

See Also

Caption (see page 2083)

37.1.51.2.14 TWebDataInput.SetFieldName Method

Pascal

```
procedure SetFieldName(Value: string);
```

C++

```
__fastcall SetFieldName(UnicodeString Value);
```

Description

This is SetFieldName, a member of class TWebDataInput.

37.1.51.2.15 TWebDataInput.SetParamName Method

Pascal

```
procedure SetParamName(Value: string);
```

C++

```
__fastcall SetParamName(UnicodeString Value);
```

Description

This is SetParamName, a member of class TWebDataInput.

37.1.51.2.16 TWebDataInput.ValidateField Method

Pascal

```
function ValidateField(DataSet: IInetXWebDataSet; AddIntf: IAddScriptElements): Boolean;
```

C++

```
__fastcall Boolean ValidateField(IInetXWebDataSet DataSet, IAddScriptElements AddIntf);
```

Description

This is ValidateField, a member of class TWebDataInput.

37.1.51.3 TWebDataInput Properties

The properties of the TWebDataInput class are listed here.

Protected Properties

	Name	Description
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Properties

	Name	Description
	FieldName (see page 2094)	Specifies the name of the field whose value the generated control represents.

37.1.51.3.1 TWebDataInput.FieldName Property

Specifies the name of the field whose value the generated control represents.

Pascal

```
property FieldName: string;
```

C++

```
__property UnicodeString FieldName;
```

Description

Use FieldName to bind the Web data input component to a field in the XML data packet. When the display control represents a field value, this is the field whose value is displayed and edited using the control. When the display control represents a parameter value, this is a field that is used to filter records that appear in data packets.

The dataset that contains the field is identified by the XMLBroker and XMLDataSetField properties of the Web item that acts as a parent to the Web data input component.

See Also

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

ParamName (see page 2094)

Setting Web Item Properties

37.1.51.3.2 TWebDataInput.ParamName Property

Specifies the name of the parameter whose value the generated control represents.

Pascal

```
property ParamName: string;
```

C++

```
__property UnicodeString ParamName;
```

Description

Use ParamName to bind the Web data input component to a parameter in the XML data packet or supply an application-specific name to the value for this control. ParamName is only used when the IsQueryField (see page 2081) method returns true. By default, the controls in a query form represent parameter values. However, they can be used to represent any named value.

If ParamName identifies one of the parameters of an XML broker, the XML broker property of the parent Web item identifies the broker that uses the parameter.

See Also

FieldName (see page 2094)

IsQueryField (see page 2081)

Params

XMLBroker (see page 2187)

Setting Web Item Properties

37.1.52 TWebDisplay Class

Class Hierarchy

Pascal

```
TWebDisplay = class(TObject);
```

C++

```
class TWebDisplay : public TObject;
```

File

MidItems

Description

This is class MidItems.TWebDisplay.

37.1.52.1 TWebDisplay Members

The following tables list the members exposed by TWebDisplay.

Protected Methods

	Name	Description
	Changed (see page 2096)	This is Changed, a member of class TWebDisplay.
V	ChangedDisplayComponent (see page 2096)	This is ChangedDisplayComponent, a member of class TWebDisplay.
V	GetDisplayComponent (see page 2096)	This is GetDisplayComponent, a member of class TWebDisplay.
	SetDisplayComponent (see page 2097)	This is SetDisplayComponent, a member of class TWebDisplay.

Public Events

	Name	Description
	OnChange (see page 2098)	This is OnChange, a member of class TWebDisplay.

Public Methods

	Name	Description
	Create (see page 2095)	This is Create, a member of class TWebDisplay.
V	Notification (see page 2097)	This is Notification, a member of class TWebDisplay.

Public Properties

	Name	Description
	DisplayComponent (see page 2097)	This is DisplayComponent, a member of class TWebDisplay.
R	Parent (see page 2097)	This is Parent, a member of class TWebDisplay.

37.1.52.2 TWebDisplay.Create Constructor

Pascal

```
constructor Create(AParent: TComponent);
```

C++

```
__fastcall TWebDisplay(TComponent * AParent);
```

Description

This is Create, a member of class TWebDisplay.

37.1.52.3 TWebDisplay Methods

The methods of the TWebDisplay class are listed here.

Protected Methods

	Name	Description
	Changed (see page 2096)	This is Changed, a member of class TWebDisplay.
V	ChangedDisplayComponent (see page 2096)	This is ChangedDisplayComponent, a member of class TWebDisplay.
V	GetDisplayComponent (see page 2096)	This is GetDisplayComponent, a member of class TWebDisplay.
	SetDisplayComponent (see page 2097)	This is SetDisplayComponent, a member of class TWebDisplay.

Public Methods

	Name	Description
V	Notification (see page 2097)	This is Notification, a member of class TWebDisplay.

37.1.52.3.1 TWebDisplay.Changed Method

Pascal

```
procedure Changed;
```

C++

```
__fastcall Changed();
```

Description

This is Changed, a member of class TWebDisplay.

37.1.52.3.2 TWebDisplay.ChangedDisplayComponent Method

Pascal

```
procedure ChangedDisplayComponent; virtual;
```

C++

```
virtual __fastcall ChangedDisplayComponent();
```

Description

This is ChangedDisplayComponent, a member of class TWebDisplay.

37.1.52.3.3 TWebDisplay.GetDisplayComponent Method

Pascal

```
function GetDisplayComponent: TComponent; virtual;
```

C++

```
virtual __fastcall TComponent * GetDisplayComponent();
```

Description

This is GetDisplayComponent, a member of class TWebDisplay.

37.1.52.3.4 TWebDisplay.Notification Method

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); virtual;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

This is Notification, a member of class TWebDisplay.

37.1.52.3.5 TWebDisplay.SetDisplayComponent Method

Pascal

```
procedure SetDisplayComponent(const Value: TComponent);
```

C++

```
__fastcall SetDisplayComponent(const TComponent * Value);
```

Description

This is SetDisplayComponent, a member of class TWebDisplay.

37.1.52.4 TWebDisplay Properties

The properties of the TWebDisplay class are listed here.

Public Properties

	Name	Description
	DisplayComponent (see page 2097)	This is DisplayComponent, a member of class TWebDisplay.
R	Parent (see page 2097)	This is Parent, a member of class TWebDisplay.

37.1.52.4.1 TWebDisplay.DisplayComponent Property

Pascal

```
property DisplayComponent: TComponent;
```

C++

```
__property TComponent * DisplayComponent;
```

Description

This is DisplayComponent, a member of class TWebDisplay.

37.1.52.4.2 TWebDisplay.Parent Property

Pascal

```
property Parent: TComponent;
```

C++

```
__property TComponent * Parent;
```

Description

This is Parent, a member of class TWebDisplay.

37.1.52.5 TWebDisplay Events

The events of the TWebDisplay class are listed here.

Public Events

	Name	Description
	OnChange (see page 2098)	This is OnChange, a member of class TWebDisplay.

37.1.52.5.1 TWebDisplay.OnChange Event

Pascal

```
property OnChange: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnChange;
```

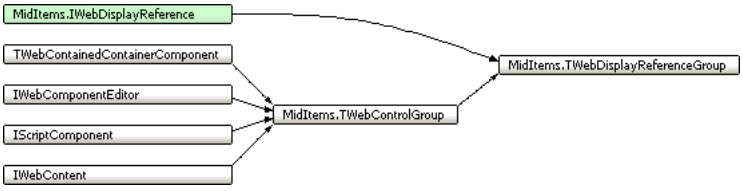
Description

This is OnChange, a member of class TWebDisplay.

37.1.53 TWebDisplayReferenceGroup Class

TWebDisplayReferenceGroup is the base class for Web items that arrange a set of controls that influence a control that displays data.

Class Hierarchy



Pascal

```
TWebDisplayReferenceGroup = class(TWebControlGroup, IWebDisplayReference);
```

C++

```
class TWebDisplayReferenceGroup : public TWebControlGroup, public IWebDisplayReference;
```

File

MidItems

Description

Use TWebDisplayReferenceGroup as a base class for Web items that generate the HTML to arrange a collection of button controls. Each TWebDisplayReferenceGroup object has an associated Web item that generates controls that display data in the final HTML document. The controls arranged by TWebDisplayReferenceGroup manipulate the information the display controls show.

In order to let other Web items use the HTML that the TWebDisplayReferenceGroup object generates, TWebDisplayReferenceGroup inherits the IWebContent interface from TWebControlGroup (see page 2064).

Do not create instances of TWebDisplayReferenceGroup. TWebDisplayReferenceGroup manages the Web items that generate buttons and the association to a display generator, but does not produce any content. Instead, use a descendant of TWebDisplayReferenceGroup such as TDataNavigator (see page 1855) (in InternetExpress applications) or TAdapterCommandGroup (in WebSnap applications).

See Also

TDataNavigator (see page 1855)

IWebContent

TAdapterCommandGroup

37.1.53.1 TWebDisplayReferenceGroup Members

The following tables list the members exposed by TWebDisplayReferenceGroup.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
	GetWebDisplayComponent (see page 2100)	This is GetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.
V	ImplAddElements (see page 2069)	Provides the underlying implementation for the AddElements (see page 2066) method.
V	ImplCanAddClass (see page 2070)	Provides the underlying implementation for the CanAddClass (see page 2067) method.
V	ImplContent (see page 2070)	Provides the underlying implementation for the Content (see page 2068) method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	Notification (see page 2101)	Forwards notification messages to all owned components.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.
V	SetWebDisplayComponent (see page 2101)	This is SetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.
R	WebDisplay (see page 2102)	This is WebDisplay, a member of class TWebDisplayReferenceGroup.
	WebDisplayComponent (see page 2102)	

Public Methods

	Name	Description
V	Create (see page 2100)	Creates an instance of TWebDisplayReferenceGroup (see page 2098).
V	Destroy (see page 2100)	Frees an instance of TWebDisplayReferenceGroup (see page 2098).

Public Properties

	Name	Description
	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.

37.1.53.2 TWebDisplayReferenceGroup.Create Constructor

Creates an instance of TWebDisplayReferenceGroup (see page 2098).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TWebDisplayReferenceGroup(TComponent * AOwner);
```

Description

Call TWebDisplayReferenceGroup (see page 2098) to instantiate TWebDisplayReferenceGroup (see page 2098) at runtime. TWebDisplayReferenceGroup (see page 2098) components added at design time are instantiated automatically.

37.1.53.3 TWebDisplayReferenceGroup.Destroy Destructor

Frees an instance of TWebDisplayReferenceGroup (see page 2098).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TWebDisplayReferenceGroup();
```

Description

Do not call Destroy directly. Instead, call Free, which checks whether the object is not nil before calling Destroy.

See Also

System::TObject::Free

37.1.53.4 TWebDisplayReferenceGroup Methods

The methods of the TWebDisplayReferenceGroup class are listed here.

Protected Methods

	Name	Description
	GetWebDisplayComponent (see page 2100)	This is GetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.
	Notification (see page 2101)	Forwards notification messages to all owned components.
	SetWebDisplayComponent (see page 2101)	This is SetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.

37.1.53.4.1 TWebDisplayReferenceGroup.GetWebDisplayComponent Method

Pascal

```
function GetWebDisplayComponent: TComponent;
```

C++

```
__fastcall TComponent * GetWebDisplayComponent();
```

Description

This is GetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.

37.1.53.4.2 TWebDisplayReferenceGroup.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

37.1.53.4.3 TWebDisplayReferenceGroup.SetWebDisplayComponent Method

Pascal

```
procedure SetWebDisplayComponent(const Value: TComponent); virtual;
```

C++

```
virtual __fastcall SetWebDisplayComponent(const TComponent * Value);
```

Description

This is SetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.

37.1.53.5 TWebDisplayReferenceGroup Properties

The properties of the TWebDisplayReferenceGroup class are listed here.

Protected Properties

	Name	Description
	WebDisplay (see page 2102)	This is WebDisplay, a member of class TWebDisplayReferenceGroup.
	WebDisplayComponent (see page 2102)	

37.1.53.5.1 TWebDisplayReferenceGroup.WebDisplay Property

Pascal

```
property WebDisplay: TWebDisplay;
```

C++

```
__property TWebDisplay WebDisplay;
```

Description

This is WebDisplay, a member of class TWebDisplayReferenceGroup.

37.1.53.5.2 TWebDisplayReferenceGroup.WebDisplayComponent Property

Pascal

```
property WebDisplayComponent: TComponent;
```

C++

```
__property TComponent * WebDisplayComponent;
```

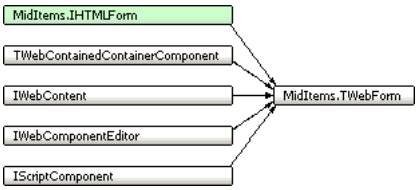
See Also

GetWebDisplayComponent (see page 2228)

37.1.54 TWebForm Class

TWebForm is the base class for Web items that create HTML forms in InternetExpress applications.

Class Hierarchy



Pascal

```
TWebForm = class(TWebContainedContainerComponent, IWebContent, IWebComponentEditor,
IScriptComponent, IHTMLForm);
```

C++

```
class TWebForm : public TWebContainedContainerComponent, public IWebContent, public
IWebComponentEditor, public IScriptComponent, public IHTMLForm;
```

File

MidItems

Description

Use TWebForm as a base class for components that generate the HTML statements defining a form that is contained in an HTML document. The list of Web items that generate content for an InternetExpress page producer can only contain descendants of TWebForm.

In order to allow the InternetExpress page producer to use the HTML generated by the TWebForm descendant, TWebForm implements the IWebContent interface. In order to allow other Web items to generate javascript that refers to form elements, TWebForm implements the IHTMLForm (see page 2211) interface.

Do not create instances of TWebForm. TWebForm relies on an abstract or, in C++ terminology, pure virtual method to generate the HTML form. Instead, create descendants of TWebForm such a TDataForm (see page 1847) or TQueryForm (see page 1971).

See Also

TQueryForm (see page 1971)

TDataForm (see page 1847)

TBasePageItemsProducer (see page 2966)

TInetXPageProducer (see page 2285)

IWebContent

IHTMLForm (see page 2211)

37.1.54.1 TWebForm Members

The following tables list the members exposed by TWebForm.

Protected Methods

	Name	Description
	AddElements (see page 2105)	Used internally to generate any javascript declarations needed by the form.
	CanAddClass (see page 2105)	Specifies whether a Web item of a specified type can be added to the WebComponents (see page 2112) property list.
	Content (see page 2106)	Returns the HTML that defines the HTML form.
V	GetDefaultWebComponents (see page 2106)	Returns nil (Delphi) or NULL (C++).
	GetHTMLFormName (see page 2107)	Returns the name of the generated HTML form.
	GetHTMLFormVarName (see page 2107)	Returns the name of the javascript variable that refers to the generated form.
V	GetLayoutAttributes (see page 2107)	Returns the object that specifies how the generated form should look.
	GetSubComponents (see page 2108)	Returns the value of the WebComponents (see page 2112) property.
	GetVisibleFields (see page 2108)	Returns the value of the VisibleFields (see page 2112) property.
V	ImplAddElements (see page 2108)	Provides the underlying implementation for the AddElements (see page 2105) method.
A	ImplCanAddClass (see page 2109)	Provides the underlying implementation for the CanAddClass (see page 2105) method.
A	ImplContent (see page 2109)	Provides the underlying implementation for the Content (see page 2106) method.
V	SetDefaultWebComponents (see page 2110)	Prevents an application from trying to assign a default set of Web items on the form.

Protected Properties

	Name	Description
R	VisibleFields (see page 2112)	Lists the Web items whose generated content appears in the HTML form.

Public Methods

	Name	Description
V	Create (see page 2104)	Creates and initializes an instance of TWebForm (see page 2102).
V	Destroy (see page 2104)	Frees an instance of TWebForm (see page 2102).

Public Properties

	Name	Description
	Custom (see page 2110)	Represents a tag option and its value in the generated FORM tag.
R	HTMLFormName (see page 2213)	Indicates the name of the generated HTML form.
R	HTMLFormVarName (see page 2213)	Indicates the name of the javascript variable that represents the HTML form.
	Style (see page 2111)	Adds style attributes to the generated form.
	StyleRule (see page 2111)	Applies a style in the page producer's style sheet to the button.
R	WebComponents (see page 2112)	Maintains a list of components that generate HTML content for the TWebForm (see page 2102) descendant.

37.1.54.2 TWebForm.Create Constructor

Creates and initializes an instance of TWebForm (see page 2102).

Pascal

```
constructor Create(AComponent: TComponent); override;
```

C++

```
virtual __fastcall TWebForm(TComponent * AComponent);
```

Description

Call Create to instantiate a TWebForm (see page 2102) descendant at runtime. TWebForm (see page 2102) descendants added at design time using the Web Page editor are instantiated automatically.

37.1.54.3 TWebForm.Destroy Destructor

Frees an instance of TWebForm (see page 2102).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TWebForm();
```

Description

Destroy removes the form generator from its parent's list of components that generate HTML before freeing the TWebForm (see page 2102) instance.

37.1.54.4 TWebForm Methods

The methods of the TWebForm class are listed here.

Protected Methods

	Name	Description
	AddElements (see page 2105)	Used internally to generate any javascript declarations needed by the form.

	CanAddClass (see page 2105)	Specifies whether a Web item of a specified type can be added to the WebComponents (see page 2112) property list.
	Content (see page 2106)	Returns the HTML that defines the HTML form.
V	GetDefaultWebComponents (see page 2106)	Returns nil (Delphi) or NULL (C++).
	GetHTMLFormName (see page 2107)	Returns the name of the generated HTML form.
	GetHTMLFormVarName (see page 2107)	Returns the name of the javascript variable that refers to the generated form.
V	GetLayoutAttributes (see page 2107)	Returns the object that specifies how the generated form should look.
	GetSubComponents (see page 2108)	Returns the value of the WebComponents (see page 2112) property.
	GetVisibleFields (see page 2108)	Returns the value of the VisibleFields (see page 2112) property.
V	ImplAddElements (see page 2108)	Provides the underlying implementation for the AddElements (see page 2105) method.
A	ImplCanAddClass (see page 2109)	Provides the underlying implementation for the CanAddClass (see page 2105) method.
A	ImplContent (see page 2109)	Provides the underlying implementation for the Content (see page 2106) method.
V	SetDefaultWebComponents (see page 2110)	Prevents an application from trying to assign a default set of Web items on the form.

37.1.54.4.1 TWebForm.AddElements Method

Used internally to generate any javascript declarations needed by the form.

Pascal

```
procedure AddElements(AddIntf: IAddScriptElements);
```

C++

```
__fastcall AddElements(IAddScriptElements AddIntf);
```

Description

The InternetExpress page producer calls AddElements when it generates the block of javascript declarations in its HTML document. This method allows the generated form to use javascript classes and methods.

AddElements calls on the virtual ImplAddElements (see page 2108) method, which uses the provided interface to return javascript declarations. Each TWebForm (see page 2102) descendant overrides ImplAddElements (see page 2108) to add any declarations it needs.

See Also

ImplAddElements (see page 2108)

TInetXPageProducer (see page 2285)

37.1.54.4.2 TWebForm.CanAddClass Method

Specifies whether a Web item of a specified type can be added to the WebComponents (see page 2112) property list.

Pascal

```
function CanAddClass(AParent: TComponent; AClass: TClass): Boolean;
```

C++

```
__fastcall Boolean CanAddClass(TComponent * AParent, TClass AClass);
```

Description

The Web page editor calls CanAddClass to determine what Web items can be added to supply content for the TWebForm (see page 2102) descendant. CanAddClass calls the virtual ImplCanAddClass (see page 2109) method to determine whether a class of Web item is valid.

AParent is the TWebForm (see page 2102) descendant to whose WebComponents (see page 2112) property the Web item

will be added.

AClass is the class of the proposed child Web item.

CanAddClass returns true if the Web form can use content generated by a component of type AClass, false otherwise.

See Also

ImplCanAddClass (see page 2109)

WebComponents (see page 2112)

37.1.54.4.3 TWebForm.Content Method

Returns the HTML that defines the HTML form.

Pascal

```
function Content(Options: TWebContentOptions; Layout: TLayout): string;
```

C++

```
__fastcall UnicodeString Content(TWebContentOptions Options, TLayout Layout);
```

Description

Content implements the IWebContent method of the same name. This method allows the page producer to extract the HTML defining the form.

Content calls the virtual ImplContent (see page 2109) method to provide the HTML form it returns. Each TWebForm (see page 2102) descendant overrides ImplContent (see page 2109) to generate its particular HTML form. The generated HTML can make use of any javascript declarations supplied by the AddElements (see page 2105) method.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the component should format the HTML form it generates. Typically, this parameter is nil (Delphi) or NULL (C++) for forms.

Content returns a string of HTML defining the form.

See Also

ImplContent (see page 2109)

IWebContent

AddElements (see page 2105)

37.1.54.4.4 TWebForm.GetDefaultWebComponents Method

Returns nil (Delphi) or NULL (C++).

Pascal

```
function GetDefaultWebComponents: TWebComponentList; override;
```

C++

```
virtual __fastcall TWebComponentList GetDefaultWebComponents();
```

Description

GetDefaultWebComponents is implemented by all Web items that use sub Web items when generating HTML. (TWebForm (see page 2102) uses the subitems in the WebComponents (see page 2112) property). This method returns a list of Web items that are used by default when an application does not explicitly assign child Web items.

In TWebForm (see page 2102), GetDefaultWebComponents returns nil (Delphi) or NULL (C++) because Web forms do not any set of items that appear by default.

See Also

WebComponents (see page 2112)

GetDefaultWebComponents

37.1.54.4.5 TWebForm.GetHTMLFormName Method

Returns the name of the generated HTML form.

Pascal

```
function GetHTMLFormName: string;
```

C++

```
__fastcall UnicodeString GetHTMLFormName();
```

Description

TWebForm (see page 2102) uses GetHTMLFormName internally to specify the NAME option of the generated form and to declare a javascript variable that references the form.

GetHTMLFormName returns the value of the Name property.

See Also

GetHTMLFormVarName (see page 2107)

Name

37.1.54.4.6 TWebForm.GetHTMLFormVarName Method

Returns the name of the javascript variable that refers to the generated form.

Pascal

```
function GetHTMLFormVarName: string;
```

C++

```
__fastcall UnicodeString GetHTMLFormVarName();
```

Description

TWebForm (see page 2102) uses GetHTMLFormVarName internally to declare a javascript variable that references the form. GetHTMLFormVarName returns the value of the Name property.

See Also

GetHTMLFormName (see page 2107)

Name

37.1.54.4.7 TWebForm.GetLayoutAttributes Method

Returns the object that specifies how the generated form should look.

Pascal

```
function GetLayoutAttributes: TLayoutAttributes; virtual;
```

C++

```
virtual __fastcall TLayoutAttributes GetLayoutAttributes();
```

Description

GetLayoutAttributes is used internally by Web items to allow the layout object supplied by a parent (as a parameter to the Content (see page 2106) method) to take into account layout properties of the child component.

As implemented by TWebForm (see page 2102), GetLayoutAttributes returns nil (Delphi) or NULL (C++), because the layout at the form level does not make use of layout attributes.

See Also

Content (see page 2106)

37.1.54.4.8 TWebForm.GetSubComponents Method

Returns the value of the WebComponents (see page 2112) property.

Pascal

```
function GetSubComponents: TObject;
```

C++

```
__fastcall TObject * GetSubComponents();
```

Description

GetSubComponents returns the TWebForm (see page 2102)'s list of Web items that it uses to generate HTML content. This method is part of the mechanism that fits TWebForm (see page 2102) into a hierarchy of Web items, each of which generates part of a larger HTML document.

See Also

WebComponents (see page 2112)

37.1.54.4.9 TWebForm.GetVisibleFields Method

Returns the value of the VisibleFields (see page 2112) property.

Pascal

```
function GetVisibleFields: TWebComponentList;
```

C++

```
__fastcall TWebComponentList GetVisibleFields();
```

Description

GetVisibleFields is the protected read implementation of the VisibleFields (see page 2112) property.

See Also

VisibleFields (see page 2112)

37.1.54.4.10 TWebForm.ImplAddElements Method

Provides the underlying implementation for the AddElements (see page 2105) method.

Pascal

```
procedure ImplAddElements(AddIntf: IAddScriptElements); virtual;
```

C++

```
virtual __fastcall ImplAddElements(IAddScriptElements AddIntf);
```

Description

The AddElements (see page 2105) method calls ImplAddElements to add any javascript declarations by calling the AddIntf interface. These declarations allow the generated display control to call javascript methods in response to user events by ensuring that the required methods are declared.

As implemented in TWebForm (see page 2102), ImplAddElements does nothing. TWebForm (see page 2102) descendants can override ImplAddElements to add any required declarations.

See Also

AddElements (see page 2105)

37.1.54.4.11 TWebForm.ImplCanAddClass Method

Provides the underlying implementation for the CanAddClass (see page 2105) method.

Pascal

```
function ImplCanAddClass(AParent: TComponent; AClass: TClass): Boolean; virtual; abstract;
```

C++

```
virtual __fastcall Boolean ImplCanAddClass(TComponent * AParent, TClass AClass) = 0;
```

Description

The CanAddClass (see page 2105) method calls ImpleCanAddClass to determine whether the TWebForm (see page 2102) descendant can add a component of type AClass to its WebComponents (see page 2112) property list. AParent is the TWebForm (see page 2102) instance to which the component will be added.

In TWebForm (see page 2102), ImplCanAddClass is abstract or, in C++ terminology, pure virtual. TWebForm (see page 2102) descendants must override ImplCanAddClass to indicate what types of Web Items they can use when generating their HTML forms.

See Also

CanAddClass (see page 2105)

WebComponents (see page 2112)

37.1.54.4.12 TWebForm.ImplContent Method

Provides the underlying implementation for the Content (see page 2106) method.

Pascal

```
function ImplContent(Options: TWebContentOptions; Layout: TLayout): string; virtual; abstract;
```

C++

```
virtual __fastcall UnicodeString ImplContent(TWebContentOptions Options, TLayout Layout) = 0;
```

Description

The Content (see page 2106) method calls ImplContent to generate the HTML form. ImplContent is anabstract or, in C++ terminology, pure virtual method in TWebForm (see page 2102). TWebForm (see page 2102) descendants override it to generate the HTML form.

Options contains a set of flags that limit what can be included in the generated HTML.

ParentLayout specifies a helper object that the caller supplies for determining how the TWebForm (see page 2102) should format the HTML control it generates. Typically, this parameter is nil (Delphi) or NULL (C++) for forms.

ImplContent returns an HTML form definition.

See Also

Content (see page 2106)

AddElements (see page 2105)

37.1.54.4.13 TWebForm.SetDefaultWebComponents Method

Prevents an application from trying to assign a default set of Web items on the form.

Pascal

```
procedure SetDefaultWebComponents(AList: TWebComponentList); override;
```

C++

```
virtual __fastcall SetDefaultWebComponents(TWebComponentList AList);
```

Description

SetDefaultWebComponents is implemented by all Web items that use sub Web items when generating HTML. (TWebForm (see page 2102) uses the subitems in the WebComponents (see page 2112) property). This method assigns a list of Web items that are used by default when an application does not explicitly assign child Web items.

In TWebForm (see page 2102), SetDefaultWebComponents does nothing with the list supplied as the AList parameter. This is because web forms do not have default items.

See Also

WebComponents (see page 2112)

SetDefaultWebComponents

37.1.54.5 TWebForm Properties

The properties of the TWebForm class are listed here.

Protected Properties

	Name	Description
R	VisibleFields (see page 2112)	Lists the Web items whose generated content appears in the HTML form.

Public Properties

	Name	Description
	Custom (see page 2110)	Represents a tag option and its value in the generated FORM tag.
	Style (see page 2111)	Adds style attributes to the generated form.
	StyleRule (see page 2111)	Applies a style in the page producer's style sheet to the button.
R	WebComponents (see page 2112)	Maintains a list of components that generate HTML content for the TWebForm (see page 2102) descendant.

37.1.54.5.1 TWebForm.Custom Property

Represents a tag option and its value in the generated FORM tag.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

Use Custom to represent one or more options that qualify the FORM tag generated by the TWebForm (see page 2102) descendant. Each option and its value has the form

```
OPTIONNAME=Value
```

The options specified by Custom should be separated by spaces. For example, to specify that the form includes files with the text it submits and indicate that the page returned after submitting the form goes into a separate window, use a Custom value such as the following:

```
ENCTYPE="multipart/form-data" TARGET="ResultFrame"
```

See Also

Style (see page 2111)

37.1.54.5.2 TWebForm.Style Property

Adds style attributes to the generated form.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to add a STYLE option that qualifies the FORM tag generated by the TWebForm (see page 2102) descendant. The Style option consists of one or more attribute descriptions such as

```
font-family: arial; font-weight: bold; font-size: 10px
```

These attributes are added to the generated FORM tag as the value of a STYLE option.

See Also

Custom (see page 2110)

StyleRule (see page 2111)

37.1.54.5.3 TWebForm.StyleRule Property

Applies a style in the page producer's style sheet to the button.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated form. The value of StyleRule specifies one of the user-defined style names defined in the InternetExpress page producer's Styles or StylesFile

property.

Note: To apply a style definition that is not included in the style sheet, use the Style (see page 2111) property.

See Also

Style (see page 2111)

Styles (see page 2272)

StylesFile (see page 2273)

37.1.54.5.4 TWebForm.VisibleFields Property

Lists the Web items whose generated content appears in the HTML form.

Pascal

```
property VisibleFields: TWebComponentList;
```

C++

```
__property TWebComponentList VisibleFields;
```

Description

VisibleFields is used internally when assembling the HTML form from the HTML generated by the form's Web items. For TWebForm (see page 2102), the value of VisibleFields is always the same as WebComponents (see page 2112).

Override GetVisibleFields (see page 2108) in a descendant class to provide a default set of that appear when the WebComponents (see page 2112) property is not set.

See Also

WebComponents (see page 2112)

37.1.54.5.5 TWebForm.WebComponents Property

Maintains a list of components that generate HTML content for the TWebForm (see page 2102) descendant.

Pascal

```
property WebComponents: TWebComponentList;
```

C++

```
__property TWebComponentList WebComponents;
```

Description

WebComponents lists the components that generate content for the Web form. Each of these components supports the IWebContent interface, and some can contain their own list of Web items that help them generate their content.

The type of items that can be added to the WebComponents list are restricted by the protected CanAddClass (see page 2105) method. Descendant classes override this method to indicate what types of items can be added to the list. For example, TQueryForm (see page 1971) limits the list to components that display or manipulate named values (TQueryButtons (see page 1962), TQueryFields) or layout groups that manage the layout of those items (TLayoutGroup (see page 1939)).

At design time, add components to the WebComponents property by bringing up the Web page editor for the containing TInetXPageProducer (see page 2285) component. Select the TWebForm (see page 2102) descendant in the editor, and use the editor to add child controls.

See Also

IWebContent

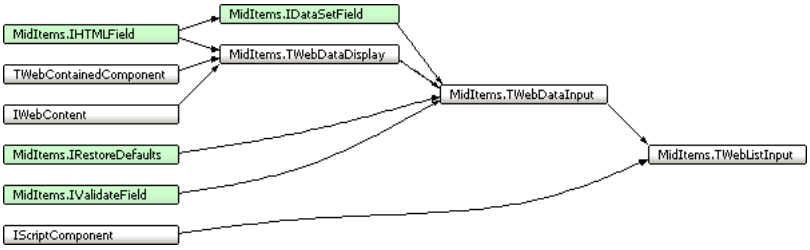
TInetXPageProducer (see page 2285)

CanAddClass (see page 2105)

37.1.55 TWebListInput Class

TWebListInput is the base class for Web items that create HTML display controls which present a fixed set of possible values.

Class Hierarchy



Pascal

```
TWebListInput = class(TWebDataInput, IScriptComponent);
```

C++

```
class TWebListInput : public TWebDataInput, public IScriptComponent;
```

File

MidItems

Description

Use TWebListInput as a base class for components that generate HTML controls for displaying a fixed set of possible values from which a user can choose. TWebListInput descendants generate controls that are organized by TDataGrid (see page 1850), TFieldGroup (see page 1869), or TQueryFieldGroup (see page 1965).

TWebListInput defines or inherits properties and methods for

Specifying a field or parameter whose value the control displays.

Specifying the possible values from which the user can choose.

Generating a caption that describes the information displayed.

Customizing the appearance of the control that displays information.

Fitting the control into the tab order of the generated HTML form.

Fitting the component into a hierarchy of Web items that collectively generate the content of an HTML document.

In order to let other Web items to use the HTML control it generates, TWebListInput inherits the IWebContent interface from TWebDataDisplay (see page 2073).

Do not create instances of TWebListInput. Instead, use a descendant of TWebListInput such as TFieldRadioGroup (see page 1874), TQueryRadioGroup (see page 1976), TFieldSelectOptions (see page 1884), TQuerySelectOptions (see page 1989), or TSelectOptionsColumn (see page 2024).

See Also

IWebContent

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

TFieldRadioGroup (see page 1874)

TQueryRadioGroup (see page 1976)

TFieldSelectOptions (see page 1884)

TQuerySelectOptions (see page 1989)

TSelectOptionsColumn (see page 2024)

37.1.55.1 TWebListInput Members

The following tables list the members exposed by TWebListInput.

Protected Methods

	Name	Description
	AddElements (see page 2117)	This is AddElements, a member of class TWebListInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
V	ControlContent (see page 2117)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
V	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
V	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
A	FormatInputs (see page 2117)	This is FormatInputs, a member of class TWebListInput.
V	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
V	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
V	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetSubComponents (see page 2118)	This is GetSubComponents, a member of class TWebListInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplAddElements (see page 2118)	This is ImplAddElements, a member of class TWebListInput.
V	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
V	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
V	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
V	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
V	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
V	Notification (see page 2118)	Forwards notification messages to all owned components.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.

	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetDataSet (see page 2119)	This is SetDataSet, a member of class TWebListInput.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetItems (see page 2119)	This is SetItems, a member of class TWebListInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	SetValues (see page 2119)	This is SetValues, a member of class TWebListInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
	Create (see page 2116)	Creates and initializes an instance of TWebListInput (see page 2113).
	Destroy (see page 2116)	Frees an instance of TWebListInput (see page 2113).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
	Caption (see page 2083)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2084)	Customizes the appearance of the generated control's label.
	Custom (see page 2084)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 2119)	Specifies a dataset that supplies the possible values listed in the generated control.
	FieldName (see page 2205)	This is FieldName, a member of class IDataSetField.
	FieldName (see page 2094)	Specifies the name of the field whose value the generated control represents.
	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	Items (see page 2120)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 2120)	Indicates which field in DataSet (see page 2119) supplies the possible selections for the generated control.
	Style (see page 2085)	Adds style attributes to the generated display control.
	StyleRule (see page 2086)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2086)	Determines the position of the generated control in the HTML form's tab order.
	Values (see page 2121)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 2121)	Indicates which field in DataSet (see page 2119) supplies the possible values for the generated control.

37.1.55.2 TWebListInput.Create Constructor

Creates and initializes an instance of TWebListInput (see page 2113).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TWebListInput(TComponent * AOwner);
```

Description

Call Create to instantiate TWebListInput (see page 2113) at runtime. TWebListInput (see page 2113) descendants added at design time using the Web Page editor are instantiated automatically.

37.1.55.3 TWebListInput.Destroy Destructor

Frees an instance of TWebListInput (see page 2113).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TWebListInput();
```

Description

Do not call Destroy directly in an application. Instead, use the Free method, which checks that the component is not nil, and only then calls Destroy.

See Also

System::TObject::Free

37.1.55.4 TWebListInput Methods

The methods of the TWebListInput class are listed here.

Protected Methods

	Name	Description
	AddElements (see page 2117)	This is AddElements, a member of class TWebListInput.
V	ControlContent (see page 2117)	Returns the HTML that defines the display control (without its caption).
A	FormatInputs (see page 2117)	This is FormatInputs, a member of class TWebListInput.
	GetSubComponents (see page 2118)	This is GetSubComponents, a member of class TWebListInput.
V	ImplAddElements (see page 2118)	This is ImplAddElements, a member of class TWebListInput.
V	Notification (see page 2118)	Forwards notification messages to all owned components.
	SetDataSet (see page 2119)	This is SetDataSet, a member of class TWebListInput.
	SetItems (see page 2119)	This is SetItems, a member of class TWebListInput.
	SetValues (see page 2119)	This is SetValues, a member of class TWebListInput.

37.1.55.4.1 TWebListInput.AddElements Method

Pascal

```
procedure AddElements(AddIntf: IAddScriptElements);
```

C++

```
__fastcall AddElements(IAddScriptElements AddIntf);
```

Description

This is AddElements, a member of class TWebListInput.

37.1.55.4.2 TWebListInput.ControlContent Method

Returns the HTML that defines the display control (without its caption).

Pascal

```
function ControlContent(Options: TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString ControlContent(TWebContentOptions Options);
```

Description

ControlContent is used internally to generate the HTML for the display control. This method does not generate any HTML for the control's label, which is handled separately by the FormatCaption method.

TWebDataDisplay introduces ControlContent as an abstract (pure virtual) method. Descendant classes override this method to generate the appropriate type of display component.

Options contains a set of flags that limit what can be included in the generated HTML.

ControlContent returns a string of HTML defining the display control.

See Also

ImplContent (see page 2080)

EventContent (see page 2077)

FormatCaption (see page 2077)

37.1.55.4.3 TWebListInput.FormatInputs Method

Pascal

```
function FormatInputs(ItemsStrings: TStrings; ValuesStrings: TStrings; Options:
TWebContentOptions): string; virtual; abstract;
```

C++

```
virtual __fastcall UnicodeString FormatInputs(TStrings ItemsStrings, TStrings ValuesStrings,
TWebContentOptions Options) = 0;
```

Description

This is FormatInputs, a member of class TWebListInput.

37.1.55.4.4 TWebListInput.GetSubComponents Method

Pascal

```
function GetSubComponents: TObject;
```

C++

```
__fastcall TObject * GetSubComponents();
```

Description

This is GetSubComponents, a member of class TWebListInput.

37.1.55.4.5 TWebListInput.ImplAddElements Method

Pascal

```
procedure ImplAddElements(AddIntf: IAddScriptElements); virtual;
```

C++

```
virtual __fastcall ImplAddElements(IAddScriptElements AddIntf);
```

Description

This is ImplAddElements, a member of class TWebListInput.

37.1.55.4.6 TWebListInput.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

37.1.55.4.7 TWebListInput.SetDataSet Method

Pascal

```
procedure SetDataSet(const Value: TDataSet);
```

C++

```
__fastcall SetDataSet(const TDataSet Value);
```

Description

This is SetDataSet, a member of class TWebListInput.

37.1.55.4.8 TWebListInput.SetItems Method

Pascal

```
procedure SetItems(const Value: TStrings);
```

C++

```
__fastcall SetItems(const TStrings Value);
```

Description

This is SetItems, a member of class TWebListInput.

37.1.55.4.9 TWebListInput.SetValues Method

Pascal

```
procedure SetValues(const Value: TStrings);
```

C++

```
__fastcall SetValues(const TStrings Value);
```

Description

This is SetValues, a member of class TWebListInput.

37.1.55.5 TWebListInput Properties

The properties of the TWebListInput class are listed here.

Public Properties

	Name	Description
	DataSet (see page 2119)	Specifies a dataset that supplies the possible values listed in the generated control.
	Items (see page 2120)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 2120)	Indicates which field in DataSet (see page 2119) supplies the possible selections for the generated control.
	Values (see page 2121)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 2121)	Indicates which field in DataSet (see page 2119) supplies the possible values for the generated control.

37.1.55.5.1 TWebListInput.DataSet Property

Specifies a dataset that supplies the possible values listed in the generated control.

Pascal

```
property DataSet: TDataSet;
```

C++

```
__property TDataSet DataSet;
```

Description

Use DataSet to specify a dataset from which the component can extract the selections that appear to the user and the underlying values they represent. The ItemsField (see page 2120) property specifies the field in this dataset that lists the selections that appear in the generated control. The ValuesField (see page 2121) property specifies the field in this dataset that lists the corresponding values that are assigned to the control.

Note: If the list of possible values is not available in a dataset, use the Items (see page 2120) and Values (see page 2121) properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

ItemsField (see page 2120)

ValuesField (see page 2121)

Items (see page 2120)

Values (see page 2121)

37.1.55.5.2 TWebListInput.Items Property

Lists the possible selections that appear in the generated control.

Pascal

```
property Items: TStrings;
```

C++

```
__property TStrings Items;
```

Description

Use Items to supply a list of possible selections to the generated control. The end-user can choose from these selections to assign a value to the generated control. The value that is assigned comes from the corresponding entry in the Values (see page 2121) property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items and Values (see page 2121) properties, use the DataSet (see page 2119), ItemsField (see page 2120), and ValuesField (see page 2121) properties.

See Also

ItemsField (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.55.5.3 TWebListInput.ItemsField Property

Indicates which field in DataSet (see page 2119) supplies the possible selections for the generated control.

Pascal

```
property ItemsField: string;
```

C++

```
__property UnicodeString ItemsField;
```

Description

Use ItemsField to specify which field in the dataset specified by the DataSet (see page 2119) property contains selections that appear in the generated control. The end-user can choose from these selections to assign a value to the generated control. The value that is assigned comes from the field specified by the ValuesField (see page 2121) property.

Note: If the list of possible values is not available in a dataset, use the Items (see page 2120) and Values (see page 2121) properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.55.5.4 TWebListInput.Values Property

Lists the possible values that can be assigned to the control.

Pascal

```
property Values: TStrings;
```

C++

```
__property TStrings Values;
```

Description

Use Values to supply a list of possible values for the generated control. When the end-user chooses one of the selections listed by the Items (see page 2120) property, the control is assigned the corresponding value in the Values property.

Note: To use a dataset to supply items and values, rather than explicitly listing them in the Items (see page 2120) and Values properties, use the DataSet (see page 2119), ItemsField (see page 2120), and ValuesField (see page 2121) properties.

See Also

Items (see page 2120)

ItemsField (see page 2120)

DataSet (see page 2119)

ValuesField (see page 2121)

37.1.55.5.5 TWebListInput.ValuesField Property

Indicates which field in DataSet (see page 2119) supplies the possible values for the generated control.

Pascal

```
property ValuesField: string;
```

C++

```
__property UnicodeString ValuesField;
```

Description

Use ValuesField to specify which field in the dataset specified by the DataSet (see page 2119) property contains values that can be assigned to the generated control. When the end-user selects the corresponding value form the ItemsField (see page 2120) field, the control gets the value in the ValuesField field.

Note: If the list of possible values is not available in a dataset, use the Items (see page 2120) and Values (see page 2121) properties instead to list the selections that appear to the user and the underlying values they represent.

See Also

Items (see page 2120)

Values (see page 2121)

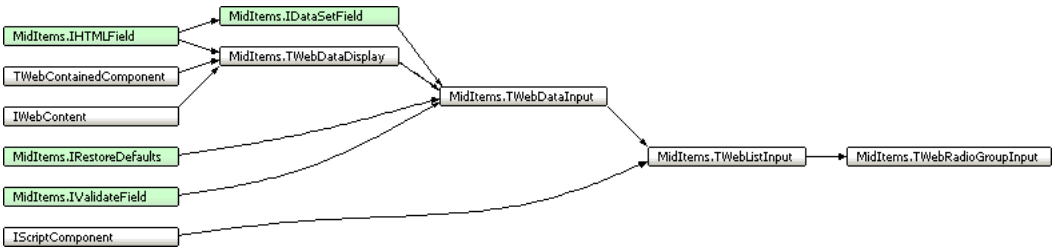
DataSet (see page 2119)

ItemsField (see page 2120)

37.1.56 TWebRadioGroupInput Class

TWebRadioGroupInput is the base class for Web items that create HTML radio group controls.

Class Hierarchy



Pascal

```
TWebRadioGroupInput = class(TWebListInput);
```

C++

```
class TWebRadioGroupInput : public TWebListInput;
```

File

MidItems

Description

Use TWebRadioGroupInput as a base class for components that generate HTML radio groups that represent field or parameter values. TWebRadioGroupInput descendants generate controls that are organized by TFieldGroup (see page 1869) or TQueryFieldGroup (see page 1965).

TWebRadioGroupInput defines or inherits properties and methods for

Specifying a field or parameter whose value the radio group displays.

Specifying the possible values from which the user can choose.

Generating a caption that describes the information displayed.

Customizing the appearance of the radio group.

Fitting the radio group into the tab order of the generated HTML form.

Fitting the component into a hierarchy of Web items that collectively generate the content of an HTML document.

In order to let other Web items to use the HTML control it generates, TWebRadioGroupInput inherits the IWebContent interface from TWebDataDisplay (see page 2073).

Do not create instances of TWebRadioGroupInput. Instead, use a descendant of TWebRadioGroupInput such as TFieldRadioGroup (see page 1874) or TQueryRadioGroup (see page 1976).

See Also

IWebContent

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

TFieldRadioGroup (see page 1874)

TQueryRadioGroup (see page 1976)

37.1.56.1 TWebRadioGroupInput Members

The following tables list the members exposed by TWebRadioGroupInput.

Protected Methods

	Name	Description
V	AddAttributes (see page 2125)	This is AddAttributes, a member of class TWebRadioGroupInput.
	AddElements (see page 2117)	This is AddElements, a member of class TWebListInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
A	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
V	ControlContent (see page 2117)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2126)	This is EventContent, a member of class TWebRadioGroupInput.
V	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
V	FormatInputs (see page 2126)	This is FormatInputs, a member of class TWebRadioGroupInput.
V	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
V	GetCaption (see page 2090)	Returns the value of the Caption property.
V	GetCheckIndex (see page 2126)	This is GetCheckIndex, a member of class TWebRadioGroupInput.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
V	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetSubComponents (see page 2118)	This is GetSubComponents, a member of class TWebListInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplAddElements (see page 2118)	This is ImplAddElements, a member of class TWebListInput.

	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
	ImplGetRowSetFieldAttributes (see page 2126)	This is ImplGetRowSetFieldAttributes, a member of class TWebRadioGroupInput.
	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	Notification (see page 2118)	Forwards notification messages to all owned components.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
	RestoreFieldDefaults (see page 2126)	This is RestoreFieldDefaults, a member of class TWebRadioGroupInput.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetDataSet (see page 2119)	This is SetDataSet, a member of class TWebListInput.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetItems (see page 2119)	This is SetItems, a member of class TWebListInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	SetValues (see page 2119)	This is SetValues, a member of class TWebListInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
	Create (see page 2125)	Creates and initializes an instance of TWebRadioGroupInput (see page 2122).
	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	Destroy (see page 2116)	Frees an instance of TWebListInput (see page 2113).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
	Caption (see page 2083)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2084)	Customizes the appearance of the generated control's label.
	Custom (see page 2084)	Represents a tag option and its value in the tag for the display control.
	DataSet (see page 2119)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayColumns (see page 2127)	Specifies the number of radio buttons that appear in each row of the control.
	DisplayWidth (see page 2127)	Specifies the width of the radio group, in characters.
	FieldName (see page 2205)	This is FieldName, a member of class IDataSetField.

	FieldName (see page 2094)	Specifies the name of the field whose value the generated control represents.
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	Items (see page 2120)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 2120)	Indicates which field in DataSet (see page 2119) supplies the possible selections for the generated control.
	ReadOnly (see page 2128)	Indicates whether end users can use the generated control to set values.
	Style (see page 2085)	Adds style attributes to the generated display control.
	StyleRule (see page 2086)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2086)	Determines the position of the generated control in the HTML form's tab order.
	Values (see page 2121)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 2121)	Indicates which field in DataSet (see page 2119) supplies the possible values for the generated control.

37.1.56.2 TWebRadioGroupInput.Create Constructor

Creates and initializes an instance of TWebRadioGroupInput (see page 2122).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TWebRadioGroupInput(TComponent * AOwner);
```

Description

Call Create to instantiate TWebRadioGroupInput (see page 2122) at runtime. TWebRadioGroupInput (see page 2122) descendants added at design time using the Web Page editor are instantiated automatically.

37.1.56.3 TWebRadioGroupInput Methods

The methods of the TWebRadioGroupInput class are listed here.

Protected Methods

	Name	Description
V	AddAttributes (see page 2125)	This is AddAttributes, a member of class TWebRadioGroupInput.
V	EventContent (see page 2126)	This is EventContent, a member of class TWebRadioGroupInput.
V	FormatInputs (see page 2126)	This is FormatInputs, a member of class TWebRadioGroupInput.
V	GetCheckIndex (see page 2126)	This is GetCheckIndex, a member of class TWebRadioGroupInput.
V	ImplGetRowSetFieldAttributes (see page 2126)	This is ImplGetRowSetFieldAttributes, a member of class TWebRadioGroupInput.
V	RestoreFieldDefaults (see page 2126)	This is RestoreFieldDefaults, a member of class TWebRadioGroupInput.

37.1.56.3.1 TWebRadioGroupInput.AddAttributes Method

Pascal

```
procedure AddAttributes(var Attrs: string); virtual;
```

C++

```
virtual __fastcall AddAttributes(UnicodeString Attrs);
```

Description

This is AddAttributes, a member of class TWebRadioGroupInput.

37.1.56.3.2 TWebRadioGroupInput.EventContent Method

Pascal

```
function EventContent(Options: TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString EventContent(TWebContentOptions Options);
```

Description

This is EventContent, a member of class TWebRadioGroupInput.

37.1.56.3.3 TWebRadioGroupInput.FormatInputs Method

Pascal

```
function FormatInputs(ItemsStrings: TStrings; ValuesStrings: TStrings; Options:
TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString FormatInputs(TStrings ItemsStrings, TStrings ValuesStrings,
TWebContentOptions Options);
```

Description

This is FormatInputs, a member of class TWebRadioGroupInput.

37.1.56.3.4 TWebRadioGroupInput.GetCheckIndex Method

Pascal

```
function GetCheckIndex(ItemsStrings: TStrings; ValuesStrings: TStrings): Integer; virtual;
```

C++

```
virtual __fastcall int GetCheckIndex(TStrings ItemsStrings, TStrings ValuesStrings);
```

Description

This is GetCheckIndex, a member of class TWebRadioGroupInput.

37.1.56.3.5 TWebRadioGroupInput.ImplGetRowSetFieldAttributes Method

Pascal

```
function ImplGetRowSetFieldAttributes(const FieldVarName: string): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplGetRowSetFieldAttributes(const UnicodeString
FieldVarName);
```

Description

This is ImplGetRowSetFieldAttributes, a member of class TWebRadioGroupInput.

37.1.56.3.6 TWebRadioGroupInput.RestoreFieldDefaults Method

Pascal

```
procedure RestoreFieldDefaults(AField: TField); override;
```

C++

```
virtual __fastcall RestoreFieldDefaults(TField AField);
```

Description

This is RestoreFieldDefaults, a member of class TWebRadioGroupInput.

37.1.56.4 TWebRadioGroupInput Properties

The properties of the TWebRadioGroupInput class are listed here.

Public Properties

	Name	Description
	DisplayColumns (see page 2127)	Specifies the number of radio buttons that appear in each row of the control.
	DisplayWidth (see page 2127)	Specifies the width of the radio group, in characters.
	ReadOnly (see page 2128)	Indicates whether end users can use the generated control to set values.

37.1.56.4.1 TWebRadioGroupInput.DisplayColumns Property

Specifies the number of radio buttons that appear in each row of the control.

Pascal

```
property DisplayColumns: Integer;
```

C++

```
__property int DisplayColumns;
```

Description

Use DisplayColumns to specify how many radio buttons can appear in a single line of the radio group. If DisplayColumns is less than the number of items, the radio group displays its buttons in multiple rows.

See Also

ItemsField (see page 2120)

Items (see page 2120)

DisplayWidth (see page 2127)

37.1.56.4.2 TWebRadioGroupInput.DisplayWidth Property

Specifies the width of the radio group, in characters.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

Description

Use DisplayWidth to limit the width of the generated control. TWebRadioGroup generates an INPUT tag of type RADIO to implement the radio group. DisplayWidth is added as a WIDTH option.

See Also

DisplayColumns (see page 2127)

Custom (see page 2084)

37.1.56.4.3 TWebRadioGroupInput.ReadOnly Property

Indicates whether end users can use the generated control to set values.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use ReadOnly to indicate whether users can change the value displayed by the radio group. TWebRadioGroup generates an INPUT tag of type RADIO to implement the radio group. ReadOnly is added as a READONLY option.

See Also

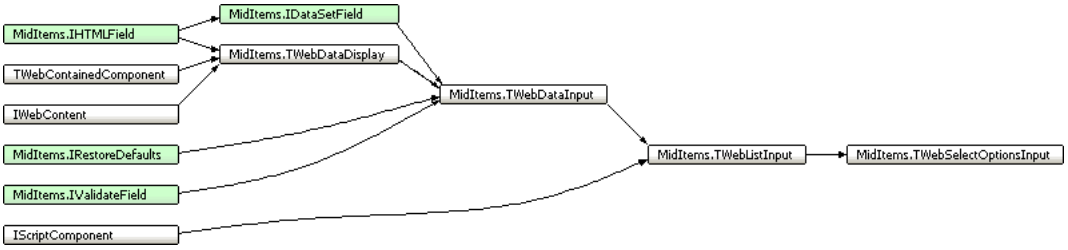
DisplayWidth (see page 2127)

Custom (see page 2084)

37.1.57 TWebSelectOptionsInput Class

TWebSelectOptionsInput is the base class for Web items that create HTML list controls.

Class Hierarchy



Pascal

```
TWebSelectOptionsInput = class(TWebListInput);
```

C++

```
class TWebSelectOptionsInput : public TWebListInput;
```

File

MidItems

Description

Use TWebSelectOptionsInput as a base class for components that generate HTML list controls that represent field or parameter values. TWebSelectOptionsInput descendants generate controls that are organized by TDataGrid (see page 1850), TFieldGroup (see page 1869), or TQueryFieldGroup (see page 1965).

TWebSelectOptionsInput defines or inherits properties and methods for

Specifying a field or parameter whose value the list box displays.

Specifying the possible values from which the user can choose.

Generating a caption that describes the information displayed.

Customizing the appearance of the list box.

Fitting the list box into the tab order of the generated HTML form.

Fitting the component into a hierarchy of Web items that collectively generate the content of an HTML document.

In order to let other Web items to use the HTML control it generates, TWebSelectOptionsInput inherits the IWebContent interface from TWebDataDisplay (see page 2073).

Do not create instances of TWebSelectOptionsInput. Instead, use a descendant of TWebSelectOptionsInput such as TFieldSelectOptions (see page 1884), TQuerySelectOptions (see page 1989), or TSelectOptionsColumn (see page 2024).

See Also

IWebContent

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

TFieldSelectOptions (see page 1884)

TQuerySelectOptions (see page 1989)

TDataGrid (see page 1850)

TSelectOptionsColumn (see page 2024)

37.1.57.1 TWebSelectOptionsInput Members

The following tables list the members exposed by TWebSelectOptionsInput.

Protected Methods

	Name	Description
V	AddAttributes (see page 2131)	This is AddAttributes, a member of class TWebSelectOptionsInput.
	AddElements (see page 2117)	This is AddElements, a member of class TWebListInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
A	ControlContent (see page 2077)	Returns the HTML that defines the display control (without its caption).
V	ControlContent (see page 2117)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2132)	This is EventContent, a member of class TWebSelectOptionsInput.
V	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
V	FormatInputs (see page 2132)	This is FormatInputs, a member of class TWebSelectOptionsInput.
V	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
V	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
V	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
V	GetSelectIndex (see page 2132)	This is GetSelectIndex, a member of class TWebSelectOptionsInput.
	GetSubComponents (see page 2118)	This is GetSubComponents, a member of class TWebListInput.

	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplAddElements (see page 2118)	This is ImplAddElements, a member of class TWebListInput.
V	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
V	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
V	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
V	ImplGetRowSetFieldAttributes (see page 2091)	This is ImplGetRowSetFieldAttributes, a member of class TWebDataInput.
V	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
V	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
V	Notification (see page 2118)	Forwards notification messages to all owned components.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
V	RestoreFieldDefaults (see page 2092)	This is RestoreFieldDefaults, a member of class TWebDataInput.
V	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
V	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetDataSet (see page 2119)	This is SetDataSet, a member of class TWebListInput.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetItems (see page 2119)	This is SetItems, a member of class TWebListInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	SetValues (see page 2119)	This is SetValues, a member of class TWebListInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
V	Create (see page 2131)	Creates and initializes an instance of TWebSelectOptionsInput (see page 2128).
V	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
V	Destroy (see page 2116)	Frees an instance of TWebListInput (see page 2113).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
V	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
V	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
	Caption (see page 2083)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2084)	Customizes the appearance of the generated control's label.
	Custom (see page 2084)	Represents a tag option and its value in the tag for the display control.

	DataSet (see page 2119)	Specifies a dataset that supplies the possible values listed in the generated control.
	DisplayRows (see page 2132)	Specifies the number of rows that appear in the generated list box.
	FieldName (see page 2205)	This is FieldName, a member of class IDataSetField.
	FieldName (see page 2094)	Specifies the name of the field whose value the generated control represents.
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	Items (see page 2120)	Lists the possible selections that appear in the generated control.
	ItemsField (see page 2120)	Indicates which field in DataSet (see page 2119) supplies the possible selections for the generated control.
	Style (see page 2085)	Adds style attributes to the generated display control.
	StyleRule (see page 2086)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2086)	Determines the position of the generated control in the HTML form's tab order.
	Values (see page 2121)	Lists the possible values that can be assigned to the control.
	ValuesField (see page 2121)	Indicates which field in DataSet (see page 2119) supplies the possible values for the generated control.

37.1.57.2 TWebSelectOptionsInput.Create Constructor

Creates and initializes an instance of TWebSelectOptionsInput (see page 2128).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TWebSelectOptionsInput(TComponent * AOwner);
```

Description

Call Create to instantiate TWebSelectOptionsInput (see page 2128) at runtime. TWebSelectOptionsInput (see page 2128) descendants added at design time using the Web Page editor are instantiated automatically.

37.1.57.3 TWebSelectOptionsInput Methods

The methods of the TWebSelectOptionsInput class are listed here.

Protected Methods

	Name	Description
V	AddAttributes (see page 2131)	This is AddAttributes, a member of class TWebSelectOptionsInput.
V	EventContent (see page 2132)	This is EventContent, a member of class TWebSelectOptionsInput.
V	FormatInputs (see page 2132)	This is FormatInputs, a member of class TWebSelectOptionsInput.
V	GetSelectIndex (see page 2132)	This is GetSelectIndex, a member of class TWebSelectOptionsInput.

37.1.57.3.1 TWebSelectOptionsInput.AddAttributes Method

Pascal

```
procedure AddAttributes(var Attrs: string); virtual;
```

C++

```
virtual __fastcall AddAttributes(UnicodeString Attrs);
```

Description

This is AddAttributes, a member of class TWebSelectOptionsInput.

37.1.57.3.2 TWebSelectOptionsInput.EventContent Method

Pascal

```
function EventContent(Options: TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString EventContent(TWebContentOptions Options);
```

Description

This is EventContent, a member of class TWebSelectOptionsInput.

37.1.57.3.3 TWebSelectOptionsInput.FormatInputs Method

Pascal

```
function FormatInputs(ItemsStrings: TStrings; ValuesStrings: TStrings; Options:
TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString FormatInputs(TStrings ItemsStrings, TStrings ValuesStrings,
TWebContentOptions Options);
```

Description

This is FormatInputs, a member of class TWebSelectOptionsInput.

37.1.57.3.4 TWebSelectOptionsInput.GetSelectIndex Method

Pascal

```
function GetSelectIndex(ItemsStrings: TStrings; ValuesStrings: TStrings): Integer; virtual;
```

C++

```
virtual __fastcall int GetSelectIndex(TStrings ItemsStrings, TStrings ValuesStrings);
```

Description

This is GetSelectIndex, a member of class TWebSelectOptionsInput.

37.1.57.4 TWebSelectOptionsInput Properties

The properties of the TWebSelectOptionsInput class are listed here.

Public Properties

	Name	Description
	DisplayRows (see page 2132)	Specifies the number of rows that appear in the generated list box.

37.1.57.4.1 TWebSelectOptionsInput.DisplayRows Property

Specifies the number of rows that appear in the generated list box.

Pascal

```
property DisplayRows: Integer;
```

C++

```
__property int DisplayRows;
```

Description

Use DisplayRows to specify how many rows appear in the generated list control. If DisplayRows is less than the number of items, the generated list displays only DisplayRows options at one time, but can scroll to display the hidden items. If DisplayRows is 1, the generated control becomes a drop-down list box.

See Also

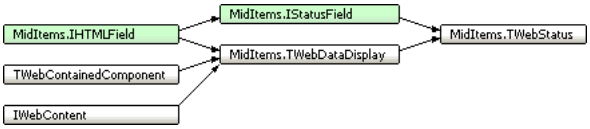
ItemsField (see page 2120)

Items (see page 2120)

37.1.58 TWebStatus Class

TWebStatus is the base class for Web items that indicate the status of records in an XML data packet.

Class Hierarchy



Pascal

```
TWebStatus = class(TWebDataDisplay, IStatusField);
```

C++

```
class TWebStatus : public TWebDataDisplay, public IStatusField;
```

File

MidItems

Description

Use TWebStatus as a base class for components that generate HTML controls which indicate whether records have been inserted or modified by the user. TWebStatus descendants generate controls that are added to TDataGrid (see page 1850) or TFieldGroup (see page 1869) to provide feedback about modifications to the underlying XML data packet.

TWebStatus defines or inherits properties and methods for

Generating a caption that describes the information displayed.

Customizing the appearance of the control.

Fitting the control into the tab order of the generated HTML form.

Fitting the component into a hierarchy of Web items that collectively generate the content of an HTML document.

In order to let other Web items to use the HTML control it generates, TWebStatus inherits the IWebContent interface from TWebDataDisplay (see page 2073).

Do not create instances of TWebStatus. Instead, use a descendant of TWebStatus such as TFieldStatus (see page 1894) or TStatusColumn (see page 2033).

See Also

IWebContent

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

TFieldStatus (see page 1894)

TStatusColumn (see page 2033)

37.1.58.1 TWebStatus Members

The following tables list the members exposed by TWebStatus.

Protected Methods

	Name	Description
	Content (see page 2076)	Returns the HTML that defines the display control.
V	ControlContent (see page 2135)	Returns the HTML that defines the display control (without its caption).
V	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
V	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
V	GetCaption (see page 2136)	Returns the value of the Caption property.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
V	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
V	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
V	ImplGetHTMLControlName (see page 2136)	Returns the name that other components can use to refer to the generated display control.
V	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
V	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.

Protected Properties

	Name	Description
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.

Public Methods

	Name	Description
V	Create (see page 2135)	Creates and initializes an instance of TWebStatus (see page 2133).
V	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	Identifier (see page 2136)	Returns the name under which record status information is stored in the XML data packet.
V	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
V	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.

Public Properties

	Name	Description
	Caption (see page 2137)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2137)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 2138)	Specifies where the label appears relative to the generated display control.
	Custom (see page 2084)	Represents a tag option and its value in the tag for the display control.
	DisplayWidth (see page 2138)	Specifies the width, in characters, of the generated HTML control.
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	Style (see page 2085)	Adds style attributes to the generated display control.
	StyleRule (see page 2086)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2086)	Determines the position of the generated control in the HTML form's tab order.

37.1.58.2 TWebStatus.Create Constructor

Creates and initializes an instance of TWebStatus (see page 2133).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TWebStatus(TComponent * AOwner);
```

Description

Call Create to instantiate TWebStatus (see page 2133) at runtime. TWebStatus (see page 2133) descendants added at design time using the Web Page editor are instantiated automatically.

37.1.58.3 TWebStatus Methods

The methods of the TWebStatus class are listed here.

Protected Methods

	Name	Description
V	ControlContent (see page 2135)	Returns the HTML that defines the display control (without its caption).
V	GetCaption (see page 2136)	Returns the value of the Caption property.
V	ImplGetHTMLControlName (see page 2136)	Returns the name that other components can use to refer to the generated display control.

Public Methods

	Name	Description
	Identifier (see page 2136)	Returns the name under which record status information is stored in the XML data packet.

37.1.58.3.1 TWebStatus.ControlContent Method

Returns the HTML that defines the display control (without its caption).

Pascal

```
function ControlContent(Options: TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString ControlContent(TWebContentOptions Options);
```

Description

ControlContent is used internally to generate the HTML for the display control. This method does not generate any HTML for the control's label, which is handled separately by the FormatCaption method.

TWebDataDisplay introduces ControlContent as an abstract (pure virtual) method. Descendant classes override this method to generate the appropriate type of display component.

Options contains a set of flags that limit what can be included in the generated HTML.

ControlContent returns a string of HTML defining the display control.

See Also

ImplContent (see page 2080)

EventContent (see page 2077)

FormatCaption (see page 2077)

37.1.58.3.2 TWebStatus.GetCaption Method

Returns the value of the Caption property.

Pascal

```
function GetCaption: string; override;
```

C++

```
virtual __fastcall UnicodeString GetCaption();
```

Description

GetCaption is the protected read implementation for the Caption property.

See Also

Caption (see page 2083)

37.1.58.3.3 TWebStatus.Identifier Method

Returns the name under which record status information is stored in the XML data packet.

Pascal

```
class function Identifier: string;
```

C++

```
__fastcall UnicodeString Identifier();
```

Description

Call Identifier to determine the name of the pseudo-field in the XML data packet that stores the status information displayed by the generated control.

37.1.58.3.4 TWebStatus.ImplGetHTMLControlName Method

Returns the name that other components can use to refer to the generated display control.

Pascal

```
function ImplGetHTMLControlName: string; override;
```

C++

```
virtual __fastcall UnicodeString ImplGetHTMLControlName();
```

Description

ImplGetHTMLControlName returns the name of the generated control. This value is assigned to a NAME option on the generated HTML tag.

ImplGetHTMLControlName returns the value of the Name property. TWebDataDisplay descendants can override this method to return a more meaningful value such as the name of the field or parameter displayed in the control.

See Also

Name

37.1.58.4 TWebStatus Properties

The properties of the TWebStatus class are listed here.

Public Properties

	Name	Description
	Caption (see page 2137)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2137)	Customizes the appearance of the generated control's label.
	CaptionPosition (see page 2138)	Specifies where the label appears relative to the generated display control.
	DisplayWidth (see page 2138)	Specifies the width, in characters, of the generated HTML control.

37.1.58.4.1 TWebStatus.Caption Property

Specifies the text of a label that indicates what information the generated control displays.

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Use Caption to specify the text of the control's label.

See Also

CaptionAttributes (see page 2084)

CaptionPosition (see page 2084)

37.1.58.4.2 TWebStatus.CaptionAttributes Property

Customizes the appearance of the generated control's label.

Pascal

```
property CaptionAttributes: TCaptionAttributes;
```

C++

```
__property TCaptionAttributes CaptionAttributes;
```

Description

Use CaptionAttributes to customize the appearance of the control's label. CaptionAttributes defines a set of options that are added to an HTML SPAN tag surrounding the value of the Caption property.

See Also

Caption (see page 2083)

CaptionPosition (see page 2084)

HeadingAttributes (see page 1817)

37.1.58.4.3 TWebStatus.CaptionPosition Property

Specifies where the label appears relative to the generated display control.

Pascal

```
property CaptionPosition: TCaptionPosition;
```

C++

```
__property TCaptionPosition CaptionPosition;
```

Description

Use CaptionPosition to indicate where the label should appear relative to the control that displays information.

Note: CaptionPosition has no effect on column controls. The position of column labels is always in the table header.

See Also

CaptionAttributes (see page 2084)

Caption (see page 2083)

37.1.58.4.4 TWebStatus.DisplayWidth Property

Specifies the width, in characters, of the generated HTML control.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

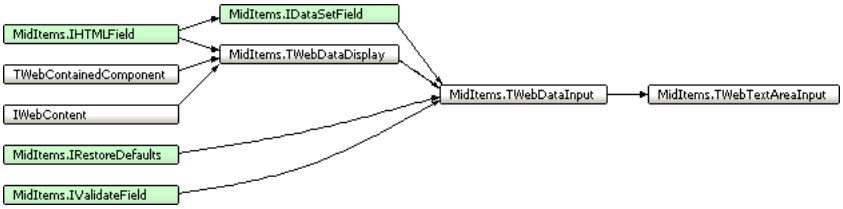
Description

Use DisplayWidth to control the width of the HTML control that displays record status information. By default, DisplayWidth is 1 because the TWebStatus (see page 2133) control values are one character (M for modified or I for inserted).

37.1.59 TWebTextAreaInput Class

TWebTextAreaInput is the base class for Web items that create HTML multi-line text controls.

Class Hierarchy



Pascal

```
TWebTextAreaInput = class(TWebDataInput);
```

C++

```
class TWebTextAreaInput : public TWebDataInput;
```

File

MidItems

Description

Use TWebTextAreaInput as a base class for components that generate HTML controls for displaying a field or parameter value in a multi-line text control. TWebTextAreaInput descendants generate controls that are organized by TDataGrid (see page 1850), TFieldGroup (see page 1869), or TQueryFieldGroup (see page 1965).

TWebTextAreaInput defines or inherits properties and methods for

Specifying a field or parameter whose value the control displays.

Generating a caption that describes the information displayed.

Customizing the appearance of the control that displays information.

Fitting the control into the tab order of the generated HTML form.

Fitting the component into a hierarchy of Web items that collectively generate the content of an HTML document.

In order to let other Web items to use the HTML control it generates, TWebTextAreaInput inherits the IWebContent interface from TWebDataDisplay (see page 2073).

Do not create instances of TWebTextAreaInput. Instead, use a descendant of TWebTextAreaInput such as TFieldTextArea (see page 1905), TQueryTextArea (see page 2010), or TTextAreaColumn (see page 2041).

See Also

IWebContent

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

TFieldTextArea (see page 1905)

TQueryTextArea (see page 2010)

TTextAreaColumn (see page 2041)

37.1.59.1 TWebTextAreaInput Members

The following tables list the members exposed by TWebTextAreaInput.

Protected Methods

	Name	Description
	AddAttributes (see page 2142)	This is AddAttributes, a member of class TWebTextAreaInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
	ControlContent (see page 2142)	Returns the HTML that defines the display control (without its caption).
	EventContent (see page 2142)	This is EventContent, a member of class TWebTextAreaInput.
	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
	ImplGetRowSetFieldAttributes (see page 2143)	This is ImplGetRowSetFieldAttributes, a member of class TWebTextAreaInput.
	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
	RestoreFieldDefaults (see page 2143)	This is RestoreFieldDefaults, a member of class TWebTextAreaInput.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.
	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
	Create (see page 2141)	Creates and initializes an instance of TWebTextAreaInput (see page 2138).

	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
	Caption (see page 2083)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2084)	Customizes the appearance of the generated control's label.
	Custom (see page 2084)	Represents a tag option and its value in the tag for the display control.
	DisplayRows (see page 2143)	Specifies the height of the generated control.
	DisplayWidth (see page 2144)	Specifies the horizontal size of the generated control.
	FieldName (see page 2205)	This is FieldName, a member of class IDataSetField.
	FieldName (see page 2094)	Specifies the name of the field whose value the generated control represents.
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	ReadOnly (see page 2144)	Indicates whether end users can use the generated control to set values.
	Style (see page 2085)	Adds style attributes to the generated display control.
	StyleRule (see page 2086)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2086)	Determines the position of the generated control in the HTML form's tab order.
	Wrap (see page 2144)	Specifies how the generated control treats lines that are longer than the display width.

37.1.59.2 TWebTextAreaInput.Create Constructor

Creates and initializes an instance of TWebTextAreaInput (see page 2138).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TWebTextAreaInput(TComponent * AOwner);
```

Description

Call Create to instantiate TWebTextAreaInput (see page 2138) at runtime. TWebTextAreaInput (see page 2138) descendants added at design time using the Web Page editor are instantiated automatically.

37.1.59.3 TWebTextAreaInput Methods

The methods of the TWebTextAreaInput class are listed here.

Protected Methods

	Name	Description
	AddAttributes (see page 2142)	This is AddAttributes, a member of class TWebTextAreaInput.
	ControlContent (see page 2142)	Returns the HTML that defines the display control (without its caption).
	EventContent (see page 2142)	This is EventContent, a member of class TWebTextAreaInput.

	ImplGetRowSetFieldAttributes (see page 2143)	This is ImplGetRowSetFieldAttributes, a member of class TWebTextAreaInput.
	RestoreFieldDefaults (see page 2143)	This is RestoreFieldDefaults, a member of class TWebTextAreaInput.

37.1.59.3.1 TWebTextAreaInput.AddAttributes Method

Pascal

```
procedure AddAttributes(var Attrs: string); virtual;
```

C++

```
virtual __fastcall AddAttributes(UnicodeString Attrs);
```

Description

This is AddAttributes, a member of class TWebTextAreaInput.

37.1.59.3.2 TWebTextAreaInput.ControlContent Method

Returns the HTML that defines the display control (without its caption).

Pascal

```
function ControlContent(Options: TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString ControlContent(TWebContentOptions Options);
```

Description

ControlContent is used internally to generate the HTML for the display control. This method does not generate any HTML for the control's label, which is handled separately by the FormatCaption method.

TWebDataDisplay introduces ControlContent as an abstract (pure virtual) method. Descendant classes override this method to generate the appropriate type of display component.

Options contains a set of flags that limit what can be included in the generated HTML.

ControlContent returns a string of HTML defining the display control.

See Also

ImplContent (see page 2080)

EventContent (see page 2077)

FormatCaption (see page 2077)

37.1.59.3.3 TWebTextAreaInput.EventContent Method

Pascal

```
function EventContent(Options: TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString EventContent(TWebContentOptions Options);
```

Description

This is EventContent, a member of class TWebTextAreaInput.

37.1.59.3.4 TWebTextAreaInput.ImplGetRowSetFieldAttributes Method

Pascal

```
function ImplGetRowSetFieldAttributes(const FieldVarName: string): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplGetRowSetFieldAttributes(const UnicodeString
FieldVarName);
```

Description

This is ImplGetRowSetFieldAttributes, a member of class TWebTextAreaInput.

37.1.59.3.5 TWebTextAreaInput.RestoreFieldDefaults Method

Pascal

```
procedure RestoreFieldDefaults(AField: TField); override;
```

C++

```
virtual __fastcall RestoreFieldDefaults(TField AField);
```

Description

This is RestoreFieldDefaults, a member of class TWebTextAreaInput.

37.1.59.4 TWebTextAreaInput Properties

The properties of the TWebTextAreaInput class are listed here.

Public Properties

	Name	Description
	DisplayRows (see page 2143)	Specifies the height of the generated control.
	DisplayWidth (see page 2144)	Specifies the horizontal size of the generated control.
	ReadOnly (see page 2144)	Indicates whether end users can use the generated control to set values.
	Wrap (see page 2144)	Specifies how the generated control treats lines that are longer than the display width.

37.1.59.4.1 TWebTextAreaInput.DisplayRows Property

Specifies the height of the generated control.

Pascal

```
property DisplayRows: Integer;
```

C++

```
__property int DisplayRows;
```

Description

Use DisplayRows to indicate the desired height of the generated HTML control. DisplayRows specifies the number of lines of text that can be displayed in the control. This value is assigned to the ROWS option of the generated TEXTAREA tag.

See Also

DisplayWidth (see page 2144)

37.1.59.4.2 TWebTextAreaInput.DisplayWidth Property

Specifies the horizontal size of the generated control.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

Description

Use DisplayWidth to indicate the desired width of the generated HTML control. DisplayWidth specifies the number of characters that can be displayed in one line of the control. This value is assigned to the COLS option of the generated TEXTAREA tag.

See Also

DisplayRows (see page 2143)

37.1.59.4.3 TWebTextAreaInput.ReadOnly Property

Indicates whether end users can use the generated control to set values.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use ReadOnly to indicate whether users can change the value displayed by the generated control. This value determines whether there is a READONLY option added to the generated TEXTAREA tag.

37.1.59.4.4 TWebTextAreaInput.Wrap Property

Specifies how the generated control treats lines that are longer than the display width.

Pascal

```
property Wrap: TTextAreaWrap;
```

C++

```
__property TTextAreaWrap Wrap;
```

Description

Use Wrap to specify how the generated control formats lines that are longer than the number of characters specified by DisplayWidth (see page 2144). TWebTextAreaInput (see page 2138) uses Wrap to add a WRAP option to the generated TEXTAREA tag.

Wrap can have any of the following values:

Value	WRAP option value	Meaning
wrOff	OFF	The control does not wrap long lines. Only newline characters that are part of the text string cause the lines in the control to wrap.

wrPhysical	PHYSICAL	The control formats its text value by wrapping long lines so that no text is hidden. The control inserts carriage returns into the text string.
wrVirtual	VIRTUAL	The control formats its text value by wrapping long lines so that no text is hidden but does not change the actual value of the text string.

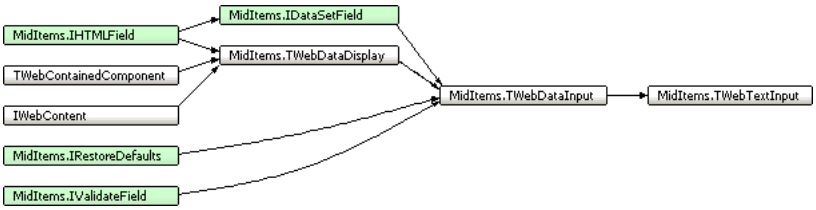
See Also

DisplayWidth (see page 2144)

37.1.60 TWebTextInput Class

TWebTextInput is the base class for Web items that create HTML single-line text controls.

Class Hierarchy



Pascal

```
TWebTextInput = class(TWebDataInput);
```

C++

```
class TWebTextInput : public TWebDataInput;
```

File

MidItems

Description

Use TWebTextInput as a base class for components that generate HTML controls for displaying a field or parameter value as a single-line text string. TWebTextInput descendants generate controls that are organized by TDataGrid (see page 1850), TFieldGroup (see page 1869), or TQueryFieldGroup (see page 1965).

TWebTextInput defines or inherits properties and methods for

Specifying a field or parameter whose value the control displays.

Generating a caption that describes the information displayed.

Customizing the appearance of the control that displays information.

Fitting the control into the tab order of the generated HTML form.

Fitting the component into a hierarchy of Web items that collectively generate the content of an HTML document.

In order to let other Web items to use the HTML control it generates, TWebTextInput inherits the IWebContent interface from TWebDataDisplay (see page 2073).

Do not create instances of TWebTextInput. Instead, use a descendant of TWebTextInput such as TFieldText (see page 1899), TQueryText (see page 2000), or TTextColumn (see page 2049).

See Also

IWebContent

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

TFieldText (see page 1899)

TQueryText (see page 2000)

TTextColumn (see page 2049)

37.1.60.1 TWebTextInput Members

The following tables list the members exposed by TWebTextInput.

Protected Methods

	Name	Description
	AddAttributes (see page 2148)	This is AddAttributes, a member of class TWebTextInput.
	Content (see page 2076)	Returns the HTML that defines the display control.
	ControlContent (see page 2148)	Returns the HTML that defines the display control (without its caption).
	EventContent (see page 2077)	This is EventContent, a member of class TWebDataDisplay.
	EventContent (see page 2089)	This is EventContent, a member of class TWebDataInput.
	FindAssociatedField (see page 2090)	This is FindAssociatedField, a member of class TWebDataInput.
	FormatCaption (see page 2077)	Returns the HTML for the display control's caption.
	GetCaption (see page 2078)	Returns the value of the Caption (see page 2083) property.
	GetCaption (see page 2090)	Returns the value of the Caption property.
	GetFieldName (see page 2090)	This is GetFieldName, a member of class TWebDataInput.
	GetHTMLControlName (see page 2078)	Returns the name that other components can use to refer to the generated display control.
	GetHTMLForm (see page 2078)	Used internally to retrieve information about the form in which the display control is defined.
	GetLayoutAttributes (see page 2079)	Returns the value of the LayoutAttributes (see page 2085) property.
	GetParamName (see page 2090)	This is GetParamName, a member of class TWebDataInput.
	GetRowSetFieldAttributes (see page 2091)	This is GetRowSetFieldAttributes, a member of class TWebDataInput.
	GetXmlDisplayName (see page 2079)	Returns the name of the javascript object that uses TWebDataDisplay (see page 2073) to generate one of its display controls. Not used on Linux.
	GetXmlRowSetName (see page 2080)	Returns the name of the rowset in the XML data packet whose data is represented by the HTML this component generates. Not used on Linux.
	ImplContent (see page 2080)	Provides the underlying implementation for the Content (see page 2076) method.
	ImplGetFieldName (see page 2091)	This is ImplGetFieldName, a member of class TWebDataInput.
	ImplGetHTMLControlName (see page 2081)	Returns the name that other components can use to refer to the generated display control.
	ImplGetHTMLControlName (see page 2091)	Returns the name that other components can use to refer to the generated display control.
	ImplGetRowSetFieldAttributes (see page 2149)	This is ImplGetRowSetFieldAttributes, a member of class TWebTextInput.
	ImplRestoreDefaults (see page 2092)	This is ImplRestoreDefaults, a member of class TWebDataInput.
	LabelContent (see page 2082)	Returns the HTML for the display control's caption.
	RestoreDefaults (see page 2092)	Returns the Web item to a default state.
	RestoreFieldDefaults (see page 2149)	This is RestoreFieldDefaults, a member of class TWebTextInput.
	SetCaption (see page 2082)	Sets the value of the Caption (see page 2083) property.
	SetCaption (see page 2092)	Sets the value of the Caption property.
	SetCaptionAttributes (see page 2083)	Sets the value of the CaptionAttributes (see page 2084) property.

	SetFieldName (see page 2093)	This is SetFieldName, a member of class TWebDataInput.
	SetParamName (see page 2093)	This is SetParamName, a member of class TWebDataInput.
	ValidateField (see page 2093)	This is ValidateField, a member of class TWebDataInput.

Protected Properties

	Name	Description
	CaptionPosition (see page 2084)	Specifies where the label appears relative to the generated display control.
R	LayoutAttributes (see page 2085)	Encodes attributes that describe how the generated control should look.
	ParamName (see page 2094)	Specifies the name of the parameter whose value the generated control represents.

Public Methods

	Name	Description
V	Create (see page 2147)	Creates and initializes an instance of TWebTextInput (see page 2145).
V	Destroy (see page 2075)	Frees an instance of TWebDataDisplay (see page 2073).
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
V	IsColumn (see page 2081)	Indicates whether the component generates the controls that appear in the column of a grid.
V	IsQueryField (see page 2081)	Indicates whether the component generates the controls that appear in a query form. Not used on Linux.
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

Public Properties

	Name	Description
	Caption (see page 2083)	Specifies the text of a label that indicates what information the generated control displays.
	CaptionAttributes (see page 2084)	Customizes the appearance of the generated control's label.
	Custom (see page 2149)	Represents a tag option and its value in the tag for the display control.
	DisplayWidth (see page 2150)	Specifies the horizontal size of the generated control.
	FieldName (see page 2205)	This is FieldName, a member of class IDataSetField.
	FieldName (see page 2094)	Specifies the name of the field whose value the generated control represents.
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	ReadOnly (see page 2150)	Indicates whether end users can use the generated control to set values.
	Style (see page 2150)	Adds style attributes to the generated display control.
	StyleRule (see page 2151)	Applies a style in the page producer's style sheet to the generated display control.
	TabIndex (see page 2086)	Determines the position of the generated control in the HTML form's tab order.

37.1.60.2 TWebTextInput.Create Constructor

Creates and initializes an instance of TWebTextInput (see page 2145).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TWebTextInput(TComponent * AOwner);
```

Description

Call Create to instantiate a TWebTextInput (see page 2145) descendant at runtime. TWebTextInput (see page 2145) descendants added at design time using the Web Page editor are instantiated automatically.

37.1.60.3 TWebTextInput Methods

The methods of the TWebTextInput class are listed here.

Protected Methods

	Name	Description
	AddAttributes (see page 2148)	This is AddAttributes, a member of class TWebTextInput.
	ControlContent (see page 2148)	Returns the HTML that defines the display control (without its caption).
	ImplGetRowSetFieldAttributes (see page 2149)	This is ImplGetRowSetFieldAttributes, a member of class TWebTextInput.
	RestoreFieldDefaults (see page 2149)	This is RestoreFieldDefaults, a member of class TWebTextInput.

37.1.60.3.1 TWebTextInput.AddAttributes Method

Pascal

```
procedure AddAttributes(var Attrs: string); virtual;
```

C++

```
virtual __fastcall AddAttributes(UnicodeString Attrs);
```

Description

This is AddAttributes, a member of class TWebTextInput.

37.1.60.3.2 TWebTextInput.ControlContent Method

Returns the HTML that defines the display control (without its caption).

Pascal

```
function ControlContent(Options: TWebContentOptions): string; override;
```

C++

```
virtual __fastcall UnicodeString ControlContent(TWebContentOptions Options);
```

Description

ControlContent is used internally to generate the HTML for the display control. This method does not generate any HTML for the control's label, which is handled separately by the FormatCaption method.

TWebDataDisplay introduces ControlContent as an abstract (pure virtual) method. Descendant classes override this method to generate the appropriate type of display component.

Options contains a set of flags that limit what can be included in the generated HTML.

ControlContent returns a string of HTML defining the display control.

See Also

ImplContent (see page 2080)

EventContent (see page 2077)

FormatCaption (see page 2077)

37.1.60.3.3 TWebTextInput.ImplGetRowSetFieldAttributes Method

Pascal

```
function ImplGetRowSetFieldAttributes(const FieldVarName: string): string; override;
```

C++

```
virtual __fastcall UnicodeString ImplGetRowSetFieldAttributes(const UnicodeString
FieldVarName);
```

Description

This is ImplGetRowSetFieldAttributes, a member of class TWebTextInput.

37.1.60.3.4 TWebTextInput.RestoreFieldDefaults Method

Pascal

```
procedure RestoreFieldDefaults(AField: TField); override;
```

C++

```
virtual __fastcall RestoreFieldDefaults(TField AField);
```

Description

This is RestoreFieldDefaults, a member of class TWebTextInput.

37.1.60.4 TWebTextInput Properties

The properties of the TWebTextInput class are listed here.

Public Properties

	Name	Description
	Custom (see page 2149)	Represents a tag option and its value in the tag for the display control.
	DisplayWidth (see page 2150)	Specifies the horizontal size of the generated control.
	ReadOnly (see page 2150)	Indicates whether end users can use the generated control to set values.
	Style (see page 2150)	Adds style attributes to the generated display control.
	StyleRule (see page 2151)	Applies a style in the page producer's style sheet to the generated display control.

37.1.60.4.1 TWebTextInput.Custom Property

Represents a tag option and its value in the tag for the display control.

Pascal

```
property Custom: string;
```

C++

```
__property UnicodeString Custom;
```

Description

TWebDataDisplay descendants generate different HTML tags to generate the control that displays information. Use Custom to add options that qualify the tag. Each option and its value has the form

OPTIONNAME=Value

The options specified by Custom should be separated by spaces. For example, to provide an INPUT control with an accelerator

key and align text to the bottom, you can set Custom to

ACCESSKEY=A ALIGN=BOTTOM

Note: To add custom options to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

37.1.60.4.2 TWebTextInput.DisplayWidth Property

Specifies the horizontal size of the generated control.

Pascal

```
property DisplayWidth: Integer;
```

C++

```
__property int DisplayWidth;
```

Description

Use DisplayWidth to indicate the desired width of the generated HTML control. DisplayWidth specifies the number of characters that can be displayed in the control.

Note: DisplayWidth controls the size of the generated control, not the size of the text that users can enter. If the user enters a string longer than DisplayWidth characters, the text scrolls in the control.

37.1.60.4.3 TWebTextInput.ReadOnly Property

Indicates whether end users can use the generated control to set values.

Pascal

```
property ReadOnly: Boolean;
```

C++

```
__property Boolean ReadOnly;
```

Description

Use ReadOnly to indicate whether users can change the value displayed by the generated control.

37.1.60.4.4 TWebTextInput.Style Property

Adds style attributes to the generated display control.

Pascal

```
property Style: string;
```

C++

```
__property UnicodeString Style;
```

Description

Use Style to specify display characteristics that affect the generated control. The Style option consists of one or more attribute descriptions such as

font-family: arial; font-weight: bold; font-size: 10px

These attributes are added to the HTML tag that generates the control as the value of a STYLE option.

Note: To add style attributes to the control's label, use the CaptionAttributes property instead.

See Also

Custom (see page 2084)

StyleRule (see page 2086)

CaptionAttributes (see page 2084)

Setting Web Item Properties

37.1.60.4.5 TWebTextInput.StyleRule Property

Applies a style in the page producer's style sheet to the generated display control.

Pascal

```
property StyleRule: string;
```

C++

```
__property UnicodeString StyleRule;
```

Description

Use StyleRule to assign one of the user-defined styles in the HTML document's style sheet to the generated display control. The value of StyleRule specifies one of the user-defined style names defined in the page producer's Styles or StylesFile property.

To apply a style definition that is not included in the style sheet, use the Style property instead.

Note: To apply a style to the control's label, use the CaptionAttributes property instead.

See Also

Style (see page 2085)

Styles (see page 2272)

Styles

StylesFile (see page 2273)

StylesFile

CaptionAttributes (see page 2084)

37.1.61 TXMLButton Class

TXMLButton is the base class for Web items that create HTML buttons that perform some action with the XML data packet.

Class Hierarchy

Pascal

```
TXMLButton = class(TWebButton);
```

C++

```
class TXMLButton : public TWebButton;
```

File

MidItems

Description

Use TXMLButton as a base class for components that generate the HTML statements defining a specialized button control for manipulating an XML data packet. TXMLButton defines properties and methods for

Generating the HTML code to define the button control.

Associating the button with an XML broker that fetches the XML data packet and responds to update messages.

Specifying tag attributes of the HTML button control, including those defined by a style sheet.

Fitting the component into a hierarchy of Web items, each of which generates part of a larger HTML document.

In order to allow other Web items to use the HTML generated by the button control, TXMLButton inherits support for the IWebContent interface from TWebButton (see page 2058).

Do not create instances of TXMLButton. TXMLButton relies on anabstract or, in C++ terminology, pure virtual method to generate the HTML statements defining a button control. Instead, create descendants of TXMLButton, such as TApplyUpdatesButton (see page 1798).

See Also

TApplyUpdatesButton (see page 1798)

IWebContent

37.1.61.1 TXMLButton Members

The following tables list the members exposed by TXMLButton.

Protected Methods

	Name	Description
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 2154)	Returns the value of the Caption property.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
	GetXMLBroker (see page 2154)	This is GetXMLBroker, a member of class TXMLButton.
	GetXMLUseParent (see page 2154)	This is GetXMLUseParent, a member of class TXMLButton.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 2154)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetXMLBroker (see page 2155)	This is SetXMLBroker, a member of class TXMLButton.
	SetXMLUseParent (see page 2155)	This is SetXMLUseParent, a member of class TXMLButton.

Protected Properties

	Name	Description
	DefaultCaption (see page 2156)	This is DefaultCaption, a member of class TXMLButton.
R	XMLData (see page 2156)	This is XMLData, a member of class TXMLButton.

Public Methods

	Name	Description
V	Create (see page 2153)	Creates and initializes an instance of TXMLButton (see page 2151).

	Destroy (see page 2153)	Frees an instance of TXMLButton (see page 2151).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLBroker (see page 2156)	Associates the XML button with an XML broker.
	XMLUseParent (see page 2156)	Specifies whether the XML button is associated with its parent's XML broker.

37.1.61.2 TXMLButton.Create Constructor

Creates and initializes an instance of TXMLButton (see page 2151).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TXMLButton(TComponent * AOwner);
```

Description

Call Create to instantiate an XML button at runtime. Web buttons added at design time using the Web Page editor are instantiated automatically.

37.1.61.3 TXMLButton.Destroy Destructor

Frees an instance of TXMLButton (see page 2151).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TXMLButton();
```

Description

Destroy removes the XML button from its parent's list of components that generate HTML before freeing the button instance.

37.1.61.4 TXMLButton Methods

The methods of the TXMLButton class are listed here.

Protected Methods

	Name	Description
	GetCaption (see page 2154)	Returns the value of the Caption property.
	GetXMLBroker (see page 2154)	This is GetXMLBroker, a member of class TXMLButton.
	GetXMLUseParent (see page 2154)	This is GetXMLUseParent, a member of class TXMLButton.
	Notification (see page 2154)	Forwards notification messages to all owned components.
	SetXMLBroker (see page 2155)	This is SetXMLBroker, a member of class TXMLButton.
	SetXMLUseParent (see page 2155)	This is SetXMLUseParent, a member of class TXMLButton.

37.1.61.4.1 TXMLButton.GetCaption Method

Returns the value of the Caption property.

Pascal

```
function GetCaption: string; override;
```

C++

```
virtual __fastcall UnicodeString GetCaption();
```

Description

GetCaption is the protected read implementation for the Caption property.

See Also

Caption (see page 2062)

SetCaption (see page 2061)

37.1.61.4.2 TXMLButton.GetXMLBroker Method

Pascal

```
function GetXMLBroker: TXMLBroker;
```

C++

```
__fastcall TXMLBroker GetXMLBroker();
```

Description

This is GetXMLBroker, a member of class TXMLButton.

37.1.61.4.3 TXMLButton.GetXMLUseParent Method

Pascal

```
function GetXMLUseParent: Boolean;
```

C++

```
__fastcall Boolean GetXMLUseParent();
```

Description

This is GetXMLUseParent, a member of class TXMLButton.

37.1.61.4.4 TXMLButton.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification

to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

37.1.61.4.5 TXMLButton.SetXMLBroker Method

Pascal

```
procedure SetXMLBroker(const Value: TXMLBroker);
```

C++

```
__fastcall SetXMLBroker(const TXMLBroker Value);
```

Description

This is SetXMLBroker, a member of class TXMLButton.

37.1.61.4.6 TXMLButton.SetXMLUseParent Method

Pascal

```
procedure SetXMLUseParent(const Value: Boolean);
```

C++

```
__fastcall SetXMLUseParent(const Boolean Value);
```

Description

This is SetXMLUseParent, a member of class TXMLButton.

37.1.61.5 TXMLButton Properties

The properties of the TXMLButton class are listed here.

Protected Properties

	Name	Description
	DefaultCaption (see page 2156)	This is DefaultCaption, a member of class TXMLButton.
R	XMLData (see page 2156)	This is XMLData, a member of class TXMLButton.

Public Properties

	Name	Description
	XMLBroker (see page 2156)	Associates the XML button with an XML broker.
	XMLUseParent (see page 2156)	Specifies whether the XML button is associated with its parent's XML broker.

37.1.61.5.1 TXMLButton.DefaultCaption Property

Pascal

```
property DefaultCaption: string;
```

C++

```
__property UnicodeString DefaultCaption;
```

Description

This is DefaultCaption, a member of class TXMLButton.

37.1.61.5.2 TXMLButton.XMLBroker Property

Associates the XML button with an XML broker.

Pascal

```
property XMLBroker: TXMLBroker;
```

C++

```
__property TXMLBroker XMLBroker;
```

Description

Set XMLBroker to specify the XML broker whose data packets the XML button manipulates.

When XML buttons are used by TDataNavigator (see page 1855), the data navigator already establishes a connection (indirectly) to an XML broker. If the XML button uses this same XML broker, you do not need to set the XMLBroker property. Instead, setting the XMLUseParent (see page 2156) property to true (the default) automatically associates the XML button with the same broker as the one associated with the navigator.

See Also

XMLUseParent (see page 2156)

37.1.61.5.3 TXMLButton.XMLData Property

Pascal

```
property XMLData: TXMLDataParent;
```

C++

```
__property TXMLDataParent XMLData;
```

Description

This is XMLData, a member of class TXMLButton.

37.1.61.5.4 TXMLButton.XMLUseParent Property

Specifies whether the XML button is associated with its parent's XML broker.

Pascal

```
property XMLUseParent: Boolean;
```

C++

```
__property Boolean XMLUseParent;
```

Description

Use XMLUseParent to indicate how the XML button determines the XML broker whose data packets it manipulates. When XMLUseParent is false, the XMLBroker (see page 2156) property must be set to indicate the XML broker. When XMLUseParent is true, the XML broker associated with the XML button's parent determines the associated XML broker.

The XML button's parent is the Web item (typically TDataNavigator (see page 1855)) that uses the XML button to assemble its own HTML content. The parent's associated broker may be defined through an indirect association. For example, the data navigator acts on another component, such as a field group, which has an associated broker.

If the parent does not have an associated XML broker, setting XMLUseParent to true sets the XMLBroker (see page 2156) property to nil (Delphi) or NULL (C++).

See Also

XMLBroker (see page 2156)

37.1.62 TXMLData Class

TXMLData manages the relationship between an XML broker and a Web item that represents a dataset in that XML broker's data packet.

Class Hierarchy

Pascal

```
TXMLData = class(TObject);
```

C++

```
class TXMLData : public TObject;
```

File

MidItems

Description

TXMLData represents the link to an XML broker for a Web item that is bound to a dataset in the XML broker's data packet. It is the base class for TXMLDataSet (see page 2164), which allows the Web item to bind to a dataset field nested in the XML broker's top-level dataset.

TXMLData does not support the ability to bind to a nested dataset field. Rather, it introduces the underlying support needed to keep track of the XML broker for components that supply this ability.

See Also

TXMLBroker

IDataSetComponent (see page 2200)

TXMLDataSet (see page 2164)

37.1.62.1 TXMLData Members

The following tables list the members exposed by TXMLData.

Protected Methods

	Name	Description
	Changed (see page 2159)	This is Changed, a member of class TXMLData.
V	ChangedXMLBroker (see page 2159)	This is ChangedXMLBroker, a member of class TXMLData.
V	GetXMLBroker (see page 2159)	This is GetXMLBroker, a member of class TXMLData.
V	SetXMLBroker (see page 2160)	This is SetXMLBroker, a member of class TXMLData.

Public Events

	Name	Description
	OnChange (see page 2161)	Occurs when the XMLBroker (see page 2160) property changes.

Public Methods

	Name	Description
	Create (see page 2158)	Creates a new TXMLData (see page 2157) object.
V	Notification (see page 2159)	Responds to notifications when objects are created or destroyed.

Public Properties

	Name	Description
R	Parent (see page 2160)	Indicates the Web item that uses this object to represent its link to an XML broker.
	XMLBroker (see page 2160)	Specifies the XML broker that generates the dataset to which the associated Web item is bound.

37.1.62.2 TXMLData.Create Constructor

Creates a new TXMLData (see page 2157) object.

Pascal

```
constructor Create(AParent: TComponent);
```

C++

```
__fastcall TXMLData(TComponent * AParent);
```

Description

Most applications do not need to call the TXMLData (see page 2157) constructor. The Web item that uses the TXMLData (see page 2157) object to represent its link to a dataset instantiates this object.

The AParent parameter specifies the Web item that uses the TXMLData (see page 2157) object to link to an XML broker. This becomes the value of the Parent (see page 2160) property.

See Also

Parent (see page 2160)

37.1.62.3 TXMLData Methods

The methods of the TXMLData class are listed here.

Protected Methods

	Name	Description
	Changed (see page 2159)	This is Changed, a member of class TXMLData.
V	ChangedXMLBroker (see page 2159)	This is ChangedXMLBroker, a member of class TXMLData.
V	GetXMLBroker (see page 2159)	This is GetXMLBroker, a member of class TXMLData.
V	SetXMLBroker (see page 2160)	This is SetXMLBroker, a member of class TXMLData.

Public Methods

	Name	Description
	Notification (see page 2159)	Responds to notifications when objects are created or destroyed.

37.1.62.3.1 TXMLData.Changed Method

Pascal

```
procedure Changed;
```

C++

```
__fastcall Changed();
```

Description

This is Changed, a member of class TXMLData.

37.1.62.3.2 TXMLData.ChangedXMLBroker Method

Pascal

```
procedure ChangedXMLBroker; virtual;
```

C++

```
virtual __fastcall ChangedXMLBroker();
```

Description

This is ChangedXMLBroker, a member of class TXMLData.

37.1.62.3.3 TXMLData.GetXMLBroker Method

Pascal

```
function GetXMLBroker: TXMLBroker; virtual;
```

C++

```
virtual __fastcall TXMLBroker GetXMLBroker();
```

Description

This is GetXMLBroker, a member of class TXMLData.

37.1.62.3.4 TXMLData.Notification Method

Responds to notifications when objects are created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); virtual;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method. It is called automatically when objects are created or destroyed. TXMLData (see page 2157) responds to notifications by setting the XMLBroker (see page 2160) property to nil (Delphi) or NULL (C++) if it is notified that this component is about to be destroyed.

See Also

XMLBroker (see page 2160)

37.1.62.3.5 TXMLData.SetXMLBroker Method

Pascal

```
procedure SetXMLBroker(const Value: TXMLBroker); virtual;
```

C++

```
virtual __fastcall SetXMLBroker(const TXMLBroker Value);
```

Description

This is SetXMLBroker, a member of class TXMLData.

37.1.62.4 TXMLData Properties

The properties of the TXMLData class are listed here.

Public Properties

	Name	Description
	Parent (see page 2160)	Indicates the Web item that uses this object to represent its link to an XML broker.
	XMLBroker (see page 2160)	Specifies the XML broker that generates the dataset to which the associated Web item is bound.

37.1.62.4.1 TXMLData.Parent Property

Indicates the Web item that uses this object to represent its link to an XML broker.

Pascal

```
property Parent: TComponent;
```

C++

```
__property TComponent * Parent;
```

Description

Read Parent to determine the Web item that uses this object to bind to a dataset from an XML broker. The value of Parent is set by the constructor.

See Also

XMLBroker (see page 2160)

Create (see page 2158)

37.1.62.4.2 TXMLData.XMLBroker Property

Specifies the XML broker that generates the dataset to which the associated Web item is bound.

Pascal

```
property XMLBroker: TXMLBroker;
```

C++

```
__property TXMLBroker XMLBroker;
```

Description

The Web item specified by the Parent (see page 2160) property sets XMLBroker to bind to an XML broker.

When the value of XMLBroker changes, the TXMLData (see page 2157) object generates an OnChange (see page 2161) event.

See Also

Parent (see page 2160)

OnChange (see page 2161)

37.1.62.5 TXMLData Events

The events of the TXMLData class are listed here.

Public Events

	Name	Description
	OnChange (see page 2161)	Occurs when the XMLBroker (see page 2160) property changes.

37.1.62.5.1 TXMLData.OnChange Event

Occurs when the XMLBroker (see page 2160) property changes.

Pascal

```
property OnChange: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnChange;
```

Description

Write an OnChange event handler to respond immediately after the Web item specified by Parent (see page 2160) becomes bound to a different XML broker. This occurs when Parent (see page 2160) changes the XMLBroker (see page 2160) property explicitly in response to a change in its own properties, or when the XML broker object to which it is bound is destroyed.

See Also

XMLBroker (see page 2160)

Parent (see page 2160)

37.1.63 TXMLDataParent Class

Class Hierarchy

Pascal

```
TXMLDataParent = class(TXMLData);
```

C++

```
class TXMLDataParent : public TXMLData;
```

File

MidItems

Description

This is class MidItems.TXMLDataParent.

37.1.63.1 TXMLDataParent Members

The following tables list the members exposed by TXMLDataParent.

Protected Methods

	Name	Description
	Changed (see page 2159)	This is Changed, a member of class TXMLData.
V	ChangedUseParent (see page 2163)	This is ChangedUseParent, a member of class TXMLDataParent.
V	ChangedXMLBroker (see page 2163)	This is ChangedXMLBroker, a member of class TXMLDataParent.
V	GetXMLBroker (see page 2163)	This is GetXMLBroker, a member of class TXMLDataParent.
	ParentXMLBroker (see page 2163)	This is ParentXMLBroker, a member of class TXMLDataParent.
	SetUseParent (see page 2163)	This is SetUseParent, a member of class TXMLDataParent.
V	SetXMLBroker (see page 2160)	This is SetXMLBroker, a member of class TXMLData.

Public Events

	Name	Description
	OnChange (see page 2161)	Occurs when the XMLBroker (see page 2160) property changes.

Public Methods

	Name	Description
	Create (see page 2158)	Creates a new TXMLData (see page 2157) object.
V	Notification (see page 2159)	Responds to notifications when objects are created or destroyed.

Public Properties

	Name	Description
R	Parent (see page 2160)	Indicates the Web item that uses this object to represent its link to an XML broker.
	XMLBroker (see page 2160)	Specifies the XML broker that generates the dataset to which the associated Web item is bound.

Published Properties

	Name	Description
	UseParent (see page 2164)	This is UseParent, a member of class TXMLDataParent.

37.1.63.2 TXMLDataParent Methods

The methods of the TXMLDataParent class are listed here.

Protected Methods

	Name	Description
V	ChangedUseParent (see page 2163)	This is ChangedUseParent, a member of class TXMLDataParent.
V	ChangedXMLBroker (see page 2163)	This is ChangedXMLBroker, a member of class TXMLDataParent.
V	GetXMLBroker (see page 2163)	This is GetXMLBroker, a member of class TXMLDataParent.
	ParentXMLBroker (see page 2163)	This is ParentXMLBroker, a member of class TXMLDataParent.
	SetUseParent (see page 2163)	This is SetUseParent, a member of class TXMLDataParent.

37.1.63.2.1 TXMLDataParent.ChangedUseParent Method

Pascal

```
procedure ChangedUseParent; virtual;
```

C++

```
virtual __fastcall ChangedUseParent();
```

Description

This is ChangedUseParent, a member of class TXMLDataParent.

37.1.63.2.2 TXMLDataParent.ChangedXMLBroker Method

Pascal

```
procedure ChangedXMLBroker; override;
```

C++

```
virtual __fastcall ChangedXMLBroker();
```

Description

This is ChangedXMLBroker, a member of class TXMLDataParent.

37.1.63.2.3 TXMLDataParent.GetXMLBroker Method

Pascal

```
function GetXMLBroker: TXMLBroker; override;
```

C++

```
virtual __fastcall TXMLBroker GetXMLBroker();
```

Description

This is GetXMLBroker, a member of class TXMLDataParent.

37.1.63.2.4 TXMLDataParent.ParentXMLBroker Method

Pascal

```
function ParentXMLBroker: TXMLBroker;
```

C++

```
__fastcall TXMLBroker ParentXMLBroker();
```

Description

This is ParentXMLBroker, a member of class TXMLDataParent.

37.1.63.2.5 TXMLDataParent.SetUseParent Method

Pascal

```
procedure SetUseParent(const Value: Boolean);
```

C++

```
__fastcall SetUseParent(const Boolean Value);
```

Description

This is SetUseParent, a member of class TXMLDataParent.

37.1.63.3 TXMLDataParent Properties

The properties of the TXMLDataParent class are listed here.

Published Properties

	Name	Description
	UseParent (see page 2164)	This is UseParent, a member of class TXMLDataParent.

37.1.63.3.1 TXMLDataParent.UseParent Property

Pascal

```
property UseParent: Boolean;
```

C++

```
__property Boolean UseParent;
```

Description

This is UseParent, a member of class TXMLDataParent.

37.1.64 TXMLDataSet Class

TXMLDataSet manages the relationship between a Web item and the XML dataset to which it is bound.

Class Hierarchy

Pascal

```
TXMLDataSet = class(TXMLData, INotifyConnectionChange);
```

C++

```
class TXMLDataSet : public TXMLData, public INotifyConnectionChange;
```

File

MidItems

Description

TXMLDataSet represents a Web item's link to an XML dataset. The Web item can be bound to an XML broker's data packet, or to a dataset field nested in an XML broker's data packet.

See Also

TXMLBroker

IDataSetComponent (see page 2200)

TXMLDisplayGroup (see page 2176)

37.1.64.1 TXMLDataSet Members

The following tables list the members exposed by TXMLDataSet.

Protected Methods

	Name	Description
	_AddRef (see page 2167)	This is _AddRef, a member of class TXMLDataSet.
	_Release (see page 2167)	This is _Release, a member of class TXMLDataSet.
	Changed (see page 2159)	This is Changed, a member of class TXMLData.
V	ChangedDataSetField (see page 2167)	This is ChangedDataSetField, a member of class TXMLDataSet.
V	ChangedXMLBroker (see page 2167)	This is ChangedXMLBroker, a member of class TXMLDataSet.
	CompareXMLData (see page 2167)	This is CompareXMLData, a member of class TXMLDataSet.
	ConnectionChange (see page 2168)	This is ConnectionChange, a member of class TXMLDataSet.
	CreateDataSet (see page 2168)	This is CreateDataSet, a member of class TXMLDataSet.
	FindDataSet (see page 2168)	This is FindDataSet, a member of class TXMLDataSet.
	GetDataSet (see page 2168)	This is GetDataSet, a member of class TXMLDataSet.
V	GetDataSetField (see page 2169)	This is GetDataSetField, a member of class TXMLDataSet.
	GetDataSetPath (see page 2169)	This is GetDataSetPath, a member of class TXMLDataSet.
V	GetXMLBroker (see page 2159)	This is GetXMLBroker, a member of class TXMLData.
V	QueryInterface (see page 2169)	This is QueryInterface, a member of class TXMLDataSet.
	SetDataSetField (see page 2169)	This is SetDataSetField, a member of class TXMLDataSet.
V	SetXMLBroker (see page 2169)	This is SetXMLBroker, a member of class TXMLDataSet.
	WrapperDataChange (see page 2170)	This is WrapperDataChange, a member of class TXMLDataSet.

Protected Properties

	Name	Description
R	Parent (see page 2171)	Indicates the Web item that uses this object to represent its link to an XML broker.

Public Events

	Name	Description
	OnChange (see page 2161)	Occurs when the XMLBroker (see page 2160) property changes.

Public Methods

	Name	Description
	Create (see page 2165)	Creates a new TXMLDataSet (see page 2164) object.
V	Destroy (see page 2166)	TXMLDataSetFrees the memory associated with the TXMLDataSet (see page 2164) object.
V	Notification (see page 2159)	Responds to notifications when objects are created or destroyed.

Public Properties

	Name	Description
R	DataSet (see page 2170)	Provides access to an interface for working with the associated dataset.
	DataSetField (see page 2171)	Identifies a nested dataset in the associated XML broker's data packet.
R	DataSetPath (see page 2171)	Represents the parsed value of the DataSetField (see page 2171) property.
	XMLBroker (see page 2160)	Specifies the XML broker that generates the dataset to which the associated Web item is bound.

37.1.64.2 TXMLDataSet.Create Constructor

Creates a new TXMLDataSet (see page 2164) object.

Pascal

```
constructor Create(AParent: TComponent);
```

C++

```
__fastcall TXMLDataSet(TComponent * AParent);
```

Description

Most applications do not need to call the TXMLDataSet (see page 2164) constructor. The Web item that uses the TXMLDataSet (see page 2164) object to represent its link to a dataset instantiates this object.

The AParent parameter specifies the Web item that uses the TXMLDataSet (see page 2164) object to link to an XML dataset. This becomes the value of the Parent (see page 2171) property.

See Also

Parent (see page 2160)

37.1.64.3 TXMLDataSet.Destroy Destructor

TXMLDataSetFrees the memory associated with the TXMLDataSet (see page 2164) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TXMLDataSet();
```

Description

Typically, applications do not free instances of TXMLDataSet (see page 2164). Instead, this object is freed by the Web item that uses it.

37.1.64.4 TXMLDataSet Methods

The methods of the TXMLDataSet class are listed here.

Protected Methods

	Name	Description
	_AddRef (see page 2167)	This is _AddRef, a member of class TXMLDataSet.
	_Release (see page 2167)	This is _Release, a member of class TXMLDataSet.
V	ChangedDataSetField (see page 2167)	This is ChangedDataSetField, a member of class TXMLDataSet.
V	ChangedXMLBroker (see page 2167)	This is ChangedXMLBroker, a member of class TXMLDataSet.
	CompareXMLData (see page 2167)	This is CompareXMLData, a member of class TXMLDataSet.
	ConnectionChange (see page 2168)	This is ConnectionChange, a member of class TXMLDataSet.
	CreateDataSet (see page 2168)	This is CreateDataSet, a member of class TXMLDataSet.
	FindDataSet (see page 2168)	This is FindDataSet, a member of class TXMLDataSet.
	GetDataSet (see page 2168)	This is GetDataSet, a member of class TXMLDataSet.
V	GetDataSetField (see page 2169)	This is GetDataSetField, a member of class TXMLDataSet.
	GetDataSetPath (see page 2169)	This is GetDataSetPath, a member of class TXMLDataSet.
V	QueryInterface (see page 2169)	This is QueryInterface, a member of class TXMLDataSet.
	SetDataSetField (see page 2169)	This is SetDataSetField, a member of class TXMLDataSet.
V	SetXMLBroker (see page 2169)	This is SetXMLBroker, a member of class TXMLDataSet.
	WrapperDataChange (see page 2170)	This is WrapperDataChange, a member of class TXMLDataSet.

37.1.64.4.1 TXMLDataSet._AddRef Method

Pascal

```
function _AddRef: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _AddRef();
```

Description

This is _AddRef, a member of class TXMLDataSet.

37.1.64.4.2 TXMLDataSet._Release Method

Pascal

```
function _Release: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _Release();
```

Description

This is _Release, a member of class TXMLDataSet.

37.1.64.4.3 TXMLDataSet.ChangedDataSetField Method

Pascal

```
procedure ChangedDataSetField; virtual;
```

C++

```
virtual __fastcall ChangedDataSetField();
```

Description

This is ChangedDataSetField, a member of class TXMLDataSet.

37.1.64.4.4 TXMLDataSet.ChangedXMLBroker Method

Pascal

```
procedure ChangedXMLBroker; override;
```

C++

```
virtual __fastcall ChangedXMLBroker();
```

Description

This is ChangedXMLBroker, a member of class TXMLDataSet.

37.1.64.4.5 TXMLDataSet.CompareXMLData Method

Pascal

```
function CompareXMLData(XMLDataSet: TXMLDataSet): Boolean;
```

C++

```
__fastcall Boolean CompareXMLData(TXMLDataSet XMLDataSet);
```

Description

This is CompareXMLData, a member of class TXMLDataSet.

37.1.64.4.6 TXMLDataSet.ConnectionChange Method

Pascal

```
procedure ConnectionChange(Sender: TComponent; Connecting: Boolean);
```

C++

```
__fastcall ConnectionChange(TComponent * Sender, Boolean Connecting);
```

Description

This is ConnectionChange, a member of class TXMLDataSet.

37.1.64.4.7 TXMLDataSet.CreateDataSet Method

Pascal

```
function CreateDataSet: IInetXWebDataSet;
```

C++

```
__fastcall IInetXWebDataSet CreateDataSet();
```

Description

This is CreateDataSet, a member of class TXMLDataSet.

37.1.64.4.8 TXMLDataSet.FindDataSet Method

Pascal

```
function FindDataSet: IInetXWebDataSet;
```

C++

```
__fastcall IInetXWebDataSet FindDataSet();
```

Description

This is FindDataSet, a member of class TXMLDataSet.

37.1.64.4.9 TXMLDataSet.GetDataSet Method

Pascal

```
function GetDataSet: IInetXWebDataSet;
```

C++

```
__fastcall IInetXWebDataSet GetDataSet();
```

Description

This is GetDataSet, a member of class TXMLDataSet.

37.1.64.4.10 TXMLDataSet.GetDataSetField Method

Pascal

```
function GetDataSetField: string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetDataSetField();
```

Description

This is GetDataSetField, a member of class TXMLDataSet.

37.1.64.4.11 TXMLDataSet.GetDataSetPath Method

Pascal

```
function GetDataSetPath: TStrings;
```

C++

```
__fastcall TStrings GetDataSetPath();
```

Description

This is GetDataSetPath, a member of class TXMLDataSet.

37.1.64.4.12 TXMLDataSet.QueryInterface Method

Pascal

```
function QueryInterface(const IID: TGUID; out Obj): HResult; virtual; stdcall;
```

C++

```
virtual __fastcall __stdcall HRESULT QueryInterface(const TGUID IID,  Obj);
```

Description

This is QueryInterface, a member of class TXMLDataSet.

37.1.64.4.13 TXMLDataSet.SetDataSetField Method

Pascal

```
procedure SetDataSetField(const Value: string);
```

C++

```
__fastcall SetDataSetField(const UnicodeString Value);
```

Description

This is SetDataSetField, a member of class TXMLDataSet.

37.1.64.4.14 TXMLDataSet.SetXMLBroker Method

Pascal

```
procedure SetXMLBroker(const Value: TXMLBroker); override;
```

C++

```
virtual __fastcall SetXMLBroker(const TXMLBroker Value);
```

Description

This is SetXMLBroker, a member of class TXMLDataSet.

37.1.64.4.15 TXMLDataSet.WrapperDataChange Method

Pascal

```
procedure WrapperDataChange(Sender: TObject);
```

C++

```
__fastcall WrapperDataChange(TObject * Sender);
```

Description

This is WrapperDataChange, a member of class TXMLDataSet.

37.1.64.5 TXMLDataSet Properties

The properties of the TXMLDataSet class are listed here.

Protected Properties

	Name	Description
	Parent (see page 2171)	Indicates the Web item that uses this object to represent its link to an XML broker.

Public Properties

	Name	Description
	DataSet (see page 2170)	Provides access to an interface for working with the associated dataset.
	DataSetField (see page 2171)	Identifies a nested dataset in the associated XML broker's data packet.
	DataSetPath (see page 2171)	Represents the parsed value of the DataSetField (see page 2171) property.

37.1.64.5.1 TXMLDataSet.DataSet Property

Provides access to an interface for working with the associated dataset.

Pascal

```
property DataSet: IInetXWebDataSet;
```

C++

```
__property IInetXWebDataSet DataSet;
```

Description

Use DataSet to manipulate the associated dataset. DataSet is an interface that provides information about the available fields and parameters of the dataset and lets you respond when this information changes.

An advantage of using the DataSet property is that it handles the task of identifying the correct dataset within the associated XML broker's data packet. This lets you treat nested datasets in the same way that you interact with the XML broker's main dataset.

See Also

XMLBroker (see page 2160)

DataSetField (see page 2171)

37.1.64.5.2 TXMLDataSet.DataSetField Property

Identifies a nested dataset in the associated XML broker's data packet.

Pascal

```
property DataSetField: string;
```

C++

```
__property UnicodeString DataSetField;
```

Description

Use DataSetField, in conjunction with the XMLBroker (see page 2160) property, to identify the XML dataset that the TXMLDataSet (see page 2164) object represents. If the dataset is the dataset represented directly by the XML broker's data packet, DataSetField is an empty string and the XMLBroker (see page 2160) property fully describes the dataset. If the dataset is a nested detail set, DataSetField indicates what detail dataset to use from the data packet associated with XMLBroker (see page 2160).

Specify the name of the dataset field as the value of DataSetField. If the dataset is the detail set of a detail set, you can use periods to qualify dataset field names. Thus, for example, if XMLBroker (see page 2160) represents a customer dataset with a nested Orders dataset field that has its own nested LineItems field, you can specify the LineItems dataset by setting DataSetField to

```
Orders.LineItems
```

See Also

XMLBroker (see page 2160)

DataSet (see page 2170)

DataSetPath (see page 2171)

37.1.64.5.3 TXMLDataSet.DataSetPath Property

Represents the parsed value of the DataSetField (see page 2171) property.

Pascal

```
property DataSetPath: TStrings;
```

C++

```
__property TStrings DataSetPath;
```

Description

DataSetPath is a parsed version of the DataSetField (see page 2171) property, where each dataset field is a separate string in the TStrings object. The order of the strings indicates the level of nesting, where the first string names a dataset field in the XML broker's data packet, the second string names a dataset field in that nested dataset, and so on.

See Also

XMLBroker (see page 2160)

DataSetField (see page 2171)

37.1.64.5.4 TXMLDataSet.Parent Property

Indicates the Web item that uses this object to represent its link to an XML broker.

Pascal

```
property Parent: TComponent;
```

C++

```
__property TComponent * Parent;
```

Description

Read Parent to determine the Web item that uses this object to bind to a dataset from an XML broker. The value of Parent is set by the constructor.

See Also

XMLBroker (see page 2160)

Create (see page 2158)

37.1.65 TXMLDataSetParent Class

Class Hierarchy

Pascal

```
TXMLDataSetParent = class(TXMLDataSet);
```

C++

```
class TXMLDataSetParent : public TXMLDataSet;
```

File

MidItems

Description

This is class MidItems.TXMLDataSetParent.

37.1.65.1 TXMLDataSetParent Members

The following tables list the members exposed by TXMLDataSetParent.

Protected Methods

	Name	Description
	_AddRef (see page 2167)	This is _AddRef, a member of class TXMLDataSet.
	_Release (see page 2167)	This is _Release, a member of class TXMLDataSet.
	Changed (see page 2159)	This is Changed, a member of class TXMLData.
V	ChangedDataSetField (see page 2174)	This is ChangedDataSetField, a member of class TXMLDataSetParent.
V	ChangedUseParent (see page 2174)	This is ChangedUseParent, a member of class TXMLDataSetParent.
V	ChangedXMLBroker (see page 2174)	This is ChangedXMLBroker, a member of class TXMLDataSetParent.
	CompareXMLData (see page 2167)	This is CompareXMLData, a member of class TXMLDataSet.
	ConnectionChange (see page 2168)	This is ConnectionChange, a member of class TXMLDataSet.
	CreateDataSet (see page 2168)	This is CreateDataSet, a member of class TXMLDataSet.
	FindDataSet (see page 2168)	This is FindDataSet, a member of class TXMLDataSet.
	GetDataSet (see page 2168)	This is GetDataSet, a member of class TXMLDataSet.

	GetDataSetField (see page 2174)	This is GetDataSetField, a member of class TXMLDataSetParent.
	GetDataSetPath (see page 2169)	This is GetDataSetPath, a member of class TXMLDataSet.
	GetXMLBroker (see page 2174)	This is GetXMLBroker, a member of class TXMLDataSetParent.
	ParentDataSetField (see page 2175)	This is ParentDataSetField, a member of class TXMLDataSetParent.
	ParentXMLBroker (see page 2175)	This is ParentXMLBroker, a member of class TXMLDataSetParent.
	ParentXMLDataSet (see page 2175)	This is ParentXMLDataSet, a member of class TXMLDataSetParent.
	QueryInterface (see page 2169)	This is QueryInterface, a member of class TXMLDataSet.
	SetDataSetField (see page 2169)	This is SetDataSetField, a member of class TXMLDataSet.
	SetUseParent (see page 2175)	This is SetUseParent, a member of class TXMLDataSetParent.
	SetXMLBroker (see page 2160)	This is SetXMLBroker, a member of class TXMLData.
	SetXMLBroker (see page 2169)	This is SetXMLBroker, a member of class TXMLDataSet.
	WrapperDataChange (see page 2170)	This is WrapperDataChange, a member of class TXMLDataSet.

Protected Properties

	Name	Description
	Parent (see page 2171)	Indicates the Web item that uses this object to represent its link to an XML broker.
	UseParent (see page 2176)	This is UseParent, a member of class TXMLDataSetParent.

Public Events

	Name	Description
	OnChange (see page 2161)	Occurs when the XMLBroker (see page 2160) property changes.

Public Methods

	Name	Description
	Create (see page 2158)	Creates a new TXMLData (see page 2157) object.
	Create (see page 2165)	Creates a new TXMLDataSet (see page 2164) object.
	Destroy (see page 2166)	TXMLDataSetFrees the memory associated with the TXMLDataSet (see page 2164) object.
	Notification (see page 2159)	Responds to notifications when objects are created or destroyed.

Public Properties

	Name	Description
	DataSet (see page 2170)	Provides access to an interface for working with the associated dataset.
	DataSetField (see page 2171)	Identifies a nested dataset in the associated XML broker's data packet.
	DataSetPath (see page 2171)	Represents the parsed value of the DataSetField (see page 2171) property.
	Parent (see page 2160)	Indicates the Web item that uses this object to represent its link to an XML broker.
	XMLBroker (see page 2160)	Specifies the XML broker that generates the dataset to which the associated Web item is bound.

37.1.65.2 TXMLDataSetParent Methods

The methods of the TXMLDataSetParent class are listed here.

Protected Methods

	Name	Description
	ChangedDataSetField (see page 2174)	This is ChangedDataSetField, a member of class TXMLDataSetParent.
	ChangedUseParent (see page 2174)	This is ChangedUseParent, a member of class TXMLDataSetParent.
	ChangedXMLBroker (see page 2174)	This is ChangedXMLBroker, a member of class TXMLDataSetParent.
	GetDataSetField (see page 2174)	This is GetDataSetField, a member of class TXMLDataSetParent.
	GetXMLBroker (see page 2174)	This is GetXMLBroker, a member of class TXMLDataSetParent.
	ParentDataSetField (see page 2175)	This is ParentDataSetField, a member of class TXMLDataSetParent.
	ParentXMLBroker (see page 2175)	This is ParentXMLBroker, a member of class TXMLDataSetParent.
	ParentXMLDataSet (see page 2175)	This is ParentXMLDataSet, a member of class TXMLDataSetParent.
	SetUseParent (see page 2175)	This is SetUseParent, a member of class TXMLDataSetParent.

37.1.65.2.1 TXMLDataSetParent.ChangedDataSetField Method

Pascal

```
procedure ChangedDataSetField; override;
```

C++

```
virtual __fastcall ChangedDataSetField();
```

Description

This is ChangedDataSetField, a member of class TXMLDataSetParent.

37.1.65.2.2 TXMLDataSetParent.ChangedUseParent Method

Pascal

```
procedure ChangedUseParent; virtual;
```

C++

```
virtual __fastcall ChangedUseParent();
```

Description

This is ChangedUseParent, a member of class TXMLDataSetParent.

37.1.65.2.3 TXMLDataSetParent.ChangedXMLBroker Method

Pascal

```
procedure ChangedXMLBroker; override;
```

C++

```
virtual __fastcall ChangedXMLBroker();
```

Description

This is ChangedXMLBroker, a member of class TXMLDataSetParent.

37.1.65.2.4 TXMLDataSetParent.GetDataSetField Method

Pascal

```
function GetDataSetField: string; override;
```

C++

```
virtual __fastcall UnicodeString GetDataSetField();
```

Description

This is GetDataSetField, a member of class TXMLDataSetParent.

37.1.65.2.5 TXMLDataSetParent.GetXMLBroker Method

Pascal

```
function GetXMLBroker: TXMLBroker; override;
```

C++

```
virtual __fastcall TXMLBroker GetXMLBroker();
```

Description

This is GetXMLBroker, a member of class TXMLDataSetParent.

37.1.65.2.6 TXMLDataSetParent.ParentDataSetField Method

Pascal

```
function ParentDataSetField: string;
```

C++

```
__fastcall UnicodeString ParentDataSetField();
```

Description

This is ParentDataSetField, a member of class TXMLDataSetParent.

37.1.65.2.7 TXMLDataSetParent.ParentXMLBroker Method

Pascal

```
function ParentXMLBroker: TXMLBroker;
```

C++

```
__fastcall TXMLBroker ParentXMLBroker();
```

Description

This is ParentXMLBroker, a member of class TXMLDataSetParent.

37.1.65.2.8 TXMLDataSetParent.ParentXMLDataSet Method

Pascal

```
function ParentXMLDataSet: TXMLDataSet;
```

C++

```
__fastcall TXMLDataSet ParentXMLDataSet();
```

Description

This is ParentXMLDataSet, a member of class TXMLDataSetParent.

37.1.65.2.9 TXMLDataSetParent.SetUseParent Method

Pascal

```
procedure SetUseParent(const Value: Boolean);
```

C++

```
__fastcall SetUseParent(const Boolean Value);
```

Description

This is SetUseParent, a member of class TXMLDataSetParent.

37.1.65.3 TXMLDataSetParent Properties

The properties of the TXMLDataSetParent class are listed here.

Protected Properties

	Name	Description
	UseParent (see page 2176)	This is UseParent, a member of class TXMLDataSetParent.

37.1.65.3.1 TXMLDataSetParent.UseParent Property

Pascal

```
property UseParent: Boolean;
```

C++

```
__property Boolean UseParent;
```

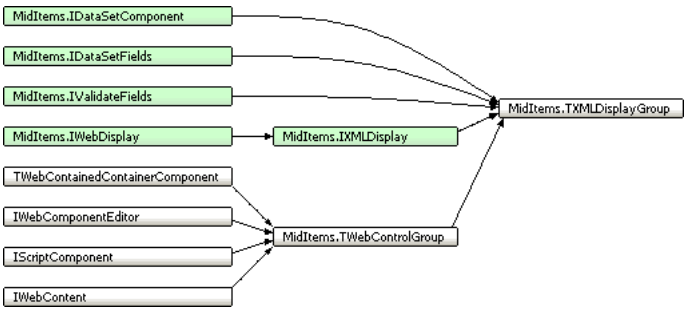
Description

This is UseParent, a member of class TXMLDataSetParent.

37.1.66 TXMLDisplayGroup Class

TXMLDisplayGroup is the base class for Web items that arrange a set of special-purpose controls in an HTML form.

Class Hierarchy



Pascal

```
TXMLDisplayGroup = class(TWebControlGroup, IDataSetComponent, IDataSetFields, IXMLDisplay,
IValidateFields);
```

C++

```
class TXMLDisplayGroup : public TWebControlGroup, public IDataSetComponent, public
IDataSetFields, public IXMLDisplay, public IValidateFields;
```

File

MidItems

Description

Use TXMLDisplayGroup as a base class for Web items that generate the HTML to arrange a collection of HTML controls. Each TXMLDisplayGroup object maintains a list of Web items that display information in the final HTML document. The XML display group arranges the controls that these items generate.

If the controls generated for TXMLDisplayGroup by its Web items represent database information, you can bind the XML display group to an XML data packet or dataset field within an XML data packet. The Web items can use that binding to determine the source of the data they display.

In order to let other Web items use the HTML that the TXMLDisplayGroup object generates, TXMLDisplayGroup inherits the IWebContent interface from TWebControlGroup (see page 2064).

Do not create instances of TXMLDisplayGroup. TXMLDisplayGroup manages the Web items that generate display controls, but does not produce any content. Instead, use a descendant of TXMLDisplayGroup that generates the HTML for its set of specialized controls such as TDataGrid (see page 1850), TFieldGroup (see page 1869) or TQueryFieldGroup (see page 1965).

See Also

TDataGrid (see page 1850)

TFieldGroup (see page 1869)

TQueryFieldGroup (see page 1965)

IWebContent

37.1.66.1 TXMLDisplayGroup Members

The following tables list the members exposed by TXMLDisplayGroup.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.
	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	CreateDefaultFields (see page 2179)	This is CreateDefaultFields, a member of class TXMLDisplayGroup.
	FindField (see page 2180)	This is FindField, a member of class TXMLDisplayGroup.
	GetDataSet (see page 2180)	Returns the value of the DataSet property.
	GetDataSetField (see page 2180)	This is GetDataSetField, a member of class TXMLDisplayGroup.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
	GetEnableValidateFields (see page 2180)	This is GetEnableValidateFields, a member of class TXMLDisplayGroup.
V	GetFieldsList (see page 2181)	This is GetFieldsList, a member of class TXMLDisplayGroup.
	GetIsMultipleRecordView (see page 2181)	This is GetIsMultipleRecordView, a member of class TXMLDisplayGroup.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
	GetVisibleFields (see page 2181)	This is GetVisibleFields, a member of class TXMLDisplayGroup.
	GetXMLBroker (see page 2181)	Returns the value of the XMLBroker property.
	GetXMLDataSet (see page 2182)	Returns the value of the XMLDataSet property.
	GetXMLDisplayName (see page 2182)	This is GetXMLDisplayName, a member of class TXMLDisplayGroup.
	GetXMLRowSetName (see page 2182)	This is GetXMLRowSetName, a member of class TXMLDisplayGroup.
	HasStatusField (see page 2182)	This is HasStatusField, a member of class TXMLDisplayGroup.
V	ImplAddElements (see page 2183)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 2070)	Provides the underlying implementation for the CanAddClass (see page 2067) method.
V	ImplContent (see page 2070)	Provides the underlying implementation for the Content (see page 2068) method.
V	ImplGetSubComponents (see page 2183)	Provides the underlying implementation for the GetSubComponents method.
V	ImplIsMultipleRecordView (see page 2183)	This is ImplIsMultipleRecordView, a member of class TXMLDisplayGroup.
	IsFieldInUse (see page 2184)	This is IsFieldInUse, a member of class TXMLDisplayGroup.
V	Notification (see page 2184)	Forwards notification messages to all owned components.
	SetDataSetField (see page 2184)	This is SetDataSetField, a member of class TXMLDisplayGroup.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.

	SetEnableValidateFields (see page 2185)	This is SetEnableValidateFields, a member of class TXMLDisplayGroup.
	SetXMLBroker (see page 2185)	This is SetXMLBroker, a member of class TXMLDisplayGroup.
	ValidateDataSet (see page 2185)	This is ValidateDataSet, a member of class TXMLDisplayGroup.
	ValidateFields (see page 2185)	This is ValidateFields, a member of class TXMLDisplayGroup.
V	XMLDataChange (see page 2185)	This is XMLDataChange, a member of class TXMLDisplayGroup.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.
	EnableValidateFields (see page 2186)	This is EnableValidateFields, a member of class TXMLDisplayGroup.
R	XMLDataSet (see page 2187)	Indicates the TXMLDataSet object that represents the dataset to which the Web item is bound.

Public Methods

	Name	Description
V	Create (see page 2178)	Creates and initializes an instance of TXMLDisplayGroup (see page 2176).
V	Destroy (see page 2178)	Frees an instance of TXMLDisplayGroup (see page 2176).

Public Properties

	Name	Description
R	DataSet (see page 2202)	Provides access to an interface for working with the associated dataset.
R	IsMultipleRecordView (see page 2227)	
R	IsMultipleRecordView (see page 2230)	
R	VisibleFields (see page 2186)	Indicates the controls displayed by the HTML that the XML display group generates.
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.
R	XMLDisplayName (see page 2231)	This is XMLDisplayName, a member of class IXMLDisplay.
R	XMLRowSetName (see page 2231)	This is XMLRowSetName, a member of class IXMLDisplay.

Published Properties

	Name	Description
	XMLBroker (see page 2187)	Binds the XML display group to an XML broker that represents a dataset.
	XMLDataSetField (see page 2188)	Binds the XML display group to a nested dataset from the XML data packet.

37.1.66.2 TXMLDisplayGroup.Create Constructor

Creates and initializes an instance of TXMLDisplayGroup (see page 2176).

Pascal

```
constructor Create(AComponent: TComponent); override;
```

C++

```
virtual __fastcall TXMLDisplayGroup(TComponent * AComponent);
```

Description

Call Create to instantiate TXMLDisplayGroup (see page 2176) at runtime. TXMLDisplayGroup (see page 2176) components added at design time using the Web Page editor are instantiated automatically.

37.1.66.3 TXMLDisplayGroup.Destroy Destructor

Frees an instance of TXMLDisplayGroup (see page 2176).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TXMLDisplayGroup();
```

Description

Do not call Destroy directly. Instead, call Free, which checks that the object is not nil before calling Destroy.

See Also

System::TObject::Free

37.1.66.4 TXMLDisplayGroup Methods

The methods of the TXMLDisplayGroup class are listed here.

Protected Methods

	Name	Description
	CreateDefaultFields (see page 2179)	This is CreateDefaultFields, a member of class TXMLDisplayGroup.
	FindField (see page 2180)	This is FindField, a member of class TXMLDisplayGroup.
	GetDataSet (see page 2180)	Returns the value of the DataSet property.
	GetDataSetField (see page 2180)	This is GetDataSetField, a member of class TXMLDisplayGroup.
	GetEnableValidateFields (see page 2180)	This is GetEnableValidateFields, a member of class TXMLDisplayGroup.
	GetFieldsList (see page 2181)	This is GetFieldsList, a member of class TXMLDisplayGroup.
	GetIsMultipleRecordView (see page 2181)	This is GetIsMultipleRecordView, a member of class TXMLDisplayGroup.
	GetVisibleFields (see page 2181)	This is GetVisibleFields, a member of class TXMLDisplayGroup.
	GetXMLBroker (see page 2181)	Returns the value of the XMLBroker property.
	GetXMLDataSet (see page 2182)	Returns the value of the XMLDataSet property.
	GetXMLDisplayName (see page 2182)	This is GetXMLDisplayName, a member of class TXMLDisplayGroup.
	GetXMLRowSetName (see page 2182)	This is GetXMLRowSetName, a member of class TXMLDisplayGroup.
	HasStatusField (see page 2182)	This is HasStatusField, a member of class TXMLDisplayGroup.
	ImplAddElements (see page 2183)	Provides the underlying implementation for the AddElements method.
	ImplGetSubComponents (see page 2183)	Provides the underlying implementation for the GetSubComponents method.
	ImplIsMultipleRecordView (see page 2183)	This is ImplIsMultipleRecordView, a member of class TXMLDisplayGroup.
	IsFieldInUse (see page 2184)	This is IsFieldInUse, a member of class TXMLDisplayGroup.
	Notification (see page 2184)	Forwards notification messages to all owned components.
	SetDataSetField (see page 2184)	This is SetDataSetField, a member of class TXMLDisplayGroup.
	SetEnableValidateFields (see page 2185)	This is SetEnableValidateFields, a member of class TXMLDisplayGroup.
	SetXMLBroker (see page 2185)	This is SetXMLBroker, a member of class TXMLDisplayGroup.
	ValidateDataSet (see page 2185)	This is ValidateDataSet, a member of class TXMLDisplayGroup.
	ValidateFields (see page 2185)	This is ValidateFields, a member of class TXMLDisplayGroup.
	XMLDataChange (see page 2185)	This is XMLDataChange, a member of class TXMLDisplayGroup.

37.1.66.4.1 TXMLDisplayGroup.CreateDefaultFields Method

Pascal

```
procedure CreateDefaultFields; virtual;
```

C++

```
virtual __fastcall CreateDefaultFields();
```

Description

This is CreateDefaultFields, a member of class TXMLDisplayGroup.

37.1.66.4.2 TXMLDisplayGroup.FindField Method

Pascal

```
function FindField(const AName: string): TComponent;
```

C++

```
__fastcall TComponent * FindField(const UnicodeString AName);
```

Description

This is FindField, a member of class TXMLDisplayGroup.

37.1.66.4.3 TXMLDisplayGroup.GetDataSet Method

Returns the value of the DataSet property.

Pascal

```
function GetDataSet: IInetXWebDataSet;
```

C++

```
__fastcall IInetXWebDataSet GetDataSet();
```

Description

GetDataSet is the read access method for the DataSet property.

See Also

DataSet (see page 2202)

37.1.66.4.4 TXMLDisplayGroup.GetDataSetField Method

Pascal

```
function GetDataSetField: string;
```

C++

```
__fastcall UnicodeString GetDataSetField();
```

Description

This is GetDataSetField, a member of class TXMLDisplayGroup.

37.1.66.4.5 TXMLDisplayGroup.GetEnableValidateFields Method

Pascal

```
function GetEnableValidateFields: Boolean;
```

C++

```
__fastcall Boolean GetEnableValidateFields();
```

Description

This is GetEnableValidateFields, a member of class TXMLDisplayGroup.

37.1.66.4.6 TXMLDisplayGroup.GetFieldsList Method

Pascal

```
procedure GetFieldsList(List: TStrings); virtual;
```

C++

```
virtual __fastcall GetFieldsList(TStrings List);
```

Description

This is GetFieldsList, a member of class TXMLDisplayGroup.

37.1.66.4.7 TXMLDisplayGroup.GetIsMultipleRecordView Method

Pascal

```
function GetIsMultipleRecordView: Boolean;
```

C++

```
__fastcall Boolean GetIsMultipleRecordView();
```

Description

This is GetIsMultipleRecordView, a member of class TXMLDisplayGroup.

37.1.66.4.8 TXMLDisplayGroup.GetVisibleFields Method

Pascal

```
function GetVisibleFields: TWebComponentList;
```

C++

```
__fastcall TWebComponentList GetVisibleFields();
```

Description

This is GetVisibleFields, a member of class TXMLDisplayGroup.

37.1.66.4.9 TXMLDisplayGroup.GetXMLBroker Method

Returns the value of the XMLBroker property.

Pascal

```
function GetXMLBroker: TXMLBroker;
```

C++

```
__fastcall TXMLBroker GetXMLBroker();
```

Description

GetXMLBroker is the read access method for the XMLBroker property.

See Also

XMLBroker (see page 2203)

37.1.66.4.10 TXMLDisplayGroup.GetXMLDataSet Method

Returns the value of the XMLDataSet property.

Pascal

```
function GetXMLDataSet: TXMLDataSet;
```

C++

```
__fastcall TXMLDataSet GetXMLDataSet();
```

Description

GetXMLDataSet is the read access method for the XMLDataSet property.

See Also

XMLDataSet (see page 2203)

37.1.66.4.11 TXMLDisplayGroup.GetXMLDisplayName Method

Pascal

```
function GetXMLDisplayName: string;
```

C++

```
__fastcall UnicodeString GetXMLDisplayName();
```

Description

This is GetXMLDisplayName, a member of class TXMLDisplayGroup.

37.1.66.4.12 TXMLDisplayGroup.GetXMLRowSetName Method

Pascal

```
function GetXMLRowSetName: string;
```

C++

```
__fastcall UnicodeString GetXMLRowSetName();
```

Description

This is GetXMLRowSetName, a member of class TXMLDisplayGroup.

37.1.66.4.13 TXMLDisplayGroup.HasStatusField Method

Pascal

```
function HasStatusField: Boolean;
```

C++

```
__fastcall Boolean HasStatusField();
```

Description

This is HasStatusField, a member of class TXMLDisplayGroup.

37.1.66.4.14 TXMLDisplayGroup.ImplAddElements Method

Provides the underlying implementation for the AddElements method.

Pascal

```
procedure ImplAddElements(AddIntf: IAddScriptElements); override;
```

C++

```
virtual __fastcall ImplAddElements(IAddScriptElements AddIntf);
```

Description

The AddElements method calls ImplAddElements to add any script declarations by calling the AddIntf interface. These declarations allow the generated display control to call script methods in response to user events by ensuring that the required methods are declared.

As implemented in TWebControlGroup, ImplAddElements does nothing. TWebControlGroup descendants override ImplAddElements to add any required declarations.

See Also

AddElements (see page 2066)

37.1.66.4.15 TXMLDisplayGroup.ImplGetSubComponents Method

Provides the underlying implementation for the GetSubComponents method.

Pascal

```
function ImplGetSubComponents: TObject; override;
```

C++

```
virtual __fastcall TObject * ImplGetSubComponents();
```

Description

The GetSubComponents method calls ImplGetSubComponents to retrieve the value of WebFieldControls as a TObject. TWebControlGroup descendants override this method if the list is not simply the value of WebFieldControls. For example, some TWebControlGroup descendants supply a default value (DefaultWebComponents) when the WebFieldControls property list is empty.

ImplGetSubComponents returns the TWebComponentList object that contains all subitems of the Web control group.

See Also

GetSubComponents (see page 2069)

DefaultWebComponents (see page 2072)

WebFieldControls (see page 2072)

37.1.66.4.16 TXMLDisplayGroup.ImplIsMultipleRecordView Method

Pascal

```
function ImplIsMultipleRecordView: Boolean; virtual;
```

C++

```
virtual __fastcall Boolean ImplIsMultipleRecordView();
```

Description

This is ImplIsMultipleRecordView, a member of class TXMLDisplayGroup.

37.1.66.4.17 TXMLDisplayGroup.IsFieldInUse Method

Pascal

```
function IsFieldInUse(AName: string): Boolean;
```

C++

```
__fastcall Boolean IsFieldInUse(UnicodeString AName);
```

Description

This is IsFieldInUse, a member of class TXMLDisplayGroup.

37.1.66.4.18 TXMLDisplayGroup.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

37.1.66.4.19 TXMLDisplayGroup.SetDataSetField Method

Pascal

```
procedure SetDataSetField(const Value: string);
```

C++

```
__fastcall SetDataSetField(const UnicodeString Value);
```

Description

This is SetDataSetField, a member of class TXMLDisplayGroup.

37.1.66.4.20 TXMLDisplayGroup.SetEnableValidateFields Method

Pascal

```
procedure SetEnableValidateFields(Value: Boolean);
```

C++

```
__fastcall SetEnableValidateFields(Boolean Value);
```

Description

This is SetEnableValidateFields, a member of class TXMLDisplayGroup.

37.1.66.4.21 TXMLDisplayGroup.SetXMLBroker Method

Pascal

```
procedure SetXMLBroker(const Value: TXMLBroker);
```

C++

```
__fastcall SetXMLBroker(const TXMLBroker Value);
```

Description

This is SetXMLBroker, a member of class TXMLDisplayGroup.

37.1.66.4.22 TXMLDisplayGroup.ValidateDataSet Method

Pascal

```
procedure ValidateDataSet;
```

C++

```
__fastcall ValidateDataSet();
```

Description

This is ValidateDataSet, a member of class TXMLDisplayGroup.

37.1.66.4.23 TXMLDisplayGroup.ValidateFields Method

Pascal

```
function ValidateFields(AddIntf: IAddScriptElements): Boolean;
```

C++

```
__fastcall Boolean ValidateFields(IAddScriptElements AddIntf);
```

Description

This is ValidateFields, a member of class TXMLDisplayGroup.

37.1.66.4.24 TXMLDisplayGroup.XMLDataChange Method

Pascal

```
procedure XMLDataChange(Sender: TObject); virtual;
```

C++

```
virtual __fastcall XMLDataChange(TObject * Sender);
```

Description

This is XMLDataChange, a member of class TXMLDisplayGroup.

37.1.66.5 TXMLDisplayGroup Properties

The properties of the TXMLDisplayGroup class are listed here.

Protected Properties

	Name	Description
	EnableValidateFields (see page 2186)	This is EnableValidateFields, a member of class TXMLDisplayGroup.
R	XMLDataSet (see page 2187)	Indicates the TXMLDataSet object that represents the dataset to which the Web item is bound.

Public Properties

	Name	Description
R	VisibleFields (see page 2186)	Indicates the controls displayed by the HTML that the XML display group generates.

Published Properties

	Name	Description
	XMLBroker (see page 2187)	Binds the XML display group to an XML broker that represents a dataset.
	XMLDataSetField (see page 2188)	Binds the XML display group to a nested dataset from the XML data packet.

37.1.66.5.1 TXMLDisplayGroup.EnableValidateFields Property

Pascal

```
property EnableValidateFields: Boolean;
```

C++

```
__property Boolean EnableValidateFields;
```

Description

This is EnableValidateFields, a member of class TXMLDisplayGroup.

37.1.66.5.2 TXMLDisplayGroup.VisibleFields Property

Indicates the controls displayed by the HTML that the XML display group generates.

Pascal

```
property VisibleFields: TWebComponentList;
```

C++

```
__property TWebComponentList VisibleFields;
```

Description

Use VisibleFields to access the components that generate HTML controls for the XML display group. If a set of Web items is explicitly assigned to the WebFieldControls (see page 2072) property, VisibleFields has the same value as WebFieldControls (see page 2072). Otherwise, VisibleFields represents a set of default Web items that generate display controls for all the fields or parameters of the associated dataset or dataset field.

When no Web items are assigned to WebFieldControls (see page 2072), the default Web items in the VisibleFields list are specific to the TXMLDisplayGroup (see page 2176) descendant.

See Also

WebFieldControls (see page 2072)

XMLBroker (see page 2187)

XMLDataSetField (see page 2188)

37.1.66.5.3 TXMLDisplayGroup.XMLBroker Property

Binds the XML display group to an XML broker that represents a dataset.

Pascal

```
property XMLBroker: TXMLBroker;
```

C++

```
__property TXMLBroker XMLBroker;
```

Description

Most XML display groups contain controls that represent database information. TDataGrid (see page 1850) and TFieldGroup (see page 1869) represent the fields from a dataset. By default, TQueryFieldGroup (see page 1965) represents the parameter values for a query or stored procedure or field values that filter a table. XMLBroker indicates the XML broker component that fetches XML data packets for the dataset whose values the controls display.

If the controls arranged by TXMLDisplayGroup (see page 2176) represent the data from a dataset nested within the XML broker's data packet, use the XMLDataSetField (see page 2188) to indicate which dataset field to use from the data packet associated with XMLBroker.

See Also

VisibleFields (see page 2186)

XMLDataSetField (see page 2188)

37.1.66.5.4 TXMLDisplayGroup.XMLDataSet Property

Indicates the TXMLDataSet object that represents the dataset to which the Web item is bound.

Pascal

```
property XMLDataSet: TXMLDataSet;
```

C++

```
__property TXMLDataSet XMLDataSet;
```

Description

XMLDataSet describes how to locate the appropriate dataset, given the XML broker that provides its XML data packet. This dataset may be the main dataset of the XML broker, or it may be a nested dataset that is represented by a dataset field within that dataset (or a dataset field nested in a dataset field, and so on).

See Also

DataSet (see page 2202)

XMLBroker (see page 2203)

37.1.66.5.5 TXMLDisplayGroup.XMLDataSetField Property

Binds the XML display group to a nested dataset from the XML data packet.

Pascal

```
property XMLDataSetField: string;
```

C++

```
__property UnicodeString XMLDataSetField;
```

Description

Most XML display groups contain controls that represent database information. TDataGrid (see page 1850) and TFieldGroup (see page 1869) represent the fields from a dataset. By default, TQueryFieldGroup (see page 1965) represents the parameter values for a query or stored procedure or field values that filter a table. If the database information comes from the dataset represented by an XML broker's data packet, XMLDataSetField is an empty string and the XMLBroker (see page 2187) property fully describes the dataset. However, if the database information comes from a nested detail set, XMLDataSetField indicates what detail dataset to use from the data packet associated with XMLBroker (see page 2187).

Specify the name of the dataset field as the value of XMLDataSetField. If the XML display group is bound to the detail set of a detail set, you can use periods to qualify dataset field names. Thus, for example, if XMLBroker (see page 2187) represents a customer dataset with a nested Orders dataset field that has its own nested LineItems field, you can bind the XML display group to the LineItems dataset by setting XMLDataSetField to

```
Orders.LineItems
```

See Also

XMLBroker (see page 2187)

VisibleFields (see page 2186)

Setting Web Item Properties

37.1.67 TXMLDisplayParent Class

Class Hierarchy

Pascal

```
TXMLDisplayParent = class(TWebDisplay);
```

C++

```
class TXMLDisplayParent : public TWebDisplay;
```

File

MidItems

Description

This is class MidItems.TXMLDisplayParent.

37.1.67.1 TXMLDisplayParent Members

The following tables list the members exposed by TXMLDisplayParent.

Protected Methods

	Name	Description
	Changed (see page 2096)	This is Changed, a member of class TWebDisplay.
V	ChangedDisplayComponent (see page 2190)	This is ChangedDisplayComponent, a member of class TXMLDisplayParent.
V	ChangedUseParent (see page 2190)	This is ChangedUseParent, a member of class TXMLDisplayParent.
V	GetDisplayComponent (see page 2190)	This is GetDisplayComponent, a member of class TXMLDisplayParent.
	SetDisplayComponent (see page 2097)	This is SetDisplayComponent, a member of class TWebDisplay.
	SetUseParent (see page 2190)	This is SetUseParent, a member of class TXMLDisplayParent.

Public Events

	Name	Description
	OnChange (see page 2098)	This is OnChange, a member of class TWebDisplay.

Public Methods

	Name	Description
	Create (see page 2189)	This is Create, a member of class TXMLDisplayParent.
V	Notification (see page 2097)	This is Notification, a member of class TWebDisplay.

Public Properties

	Name	Description
	DisplayComponent (see page 2097)	This is DisplayComponent, a member of class TWebDisplay.
R	Parent (see page 2097)	This is Parent, a member of class TWebDisplay.

Published Properties

	Name	Description
	UseParent (see page 2191)	This is UseParent, a member of class TXMLDisplayParent.

37.1.67.2 TXMLDisplayParent.Create Constructor

Pascal

```
constructor Create(AParent: TComponent);
```

C++

```
__fastcall TXMLDisplayParent(TComponent * AParent);
```

Description

This is Create, a member of class TXMLDisplayParent.

37.1.67.3 TXMLDisplayParent Methods

The methods of the TXMLDisplayParent class are listed here.

Protected Methods

	Name	Description
V	ChangedDisplayComponent (see page 2190)	This is ChangedDisplayComponent, a member of class TXMLDisplayParent.
V	ChangedUseParent (see page 2190)	This is ChangedUseParent, a member of class TXMLDisplayParent.

	GetDisplayComponent (see page 2190)	This is GetDisplayComponent, a member of class TXMLDisplayParent.
	SetUseParent (see page 2190)	This is SetUseParent, a member of class TXMLDisplayParent.

37.1.67.3.1 TXMLDisplayParent.ChangedDisplayComponent Method

Pascal

```
procedure ChangedDisplayComponent; override;
```

C++

```
virtual __fastcall ChangedDisplayComponent();
```

Description

This is ChangedDisplayComponent, a member of class TXMLDisplayParent.

37.1.67.3.2 TXMLDisplayParent.ChangedUseParent Method

Pascal

```
procedure ChangedUseParent; virtual;
```

C++

```
virtual __fastcall ChangedUseParent();
```

Description

This is ChangedUseParent, a member of class TXMLDisplayParent.

37.1.67.3.3 TXMLDisplayParent.GetDisplayComponent Method

Pascal

```
function GetDisplayComponent: TComponent; override;
```

C++

```
virtual __fastcall TComponent * GetDisplayComponent();
```

Description

This is GetDisplayComponent, a member of class TXMLDisplayParent.

37.1.67.3.4 TXMLDisplayParent.SetUseParent Method

Pascal

```
procedure SetUseParent(const Value: Boolean);
```

C++

```
__fastcall SetUseParent(const Boolean Value);
```

Description

This is SetUseParent, a member of class TXMLDisplayParent.

37.1.67.4 TXMLDisplayParent Properties

The properties of the TXMLDisplayParent class are listed here.

Published Properties

	Name	Description
	UseParent (see page 2191)	This is UseParent, a member of class TXMLDisplayParent.

37.1.67.4.1 TXMLDisplayParent.UseParent Property

Pascal

```
property UseParent: Boolean;
```

C++

```
__property Boolean UseParent;
```

Description

This is UseParent, a member of class TXMLDisplayParent.

37.1.68 TXMLDisplayReferenceButton Class

TXMLDisplayReferenceButton is the base class for Web items that generate HTML buttons for influencing controls that display data.

Class Hierarchy

Pascal

```
TXMLDisplayReferenceButton = class(TWebButton);
```

C++

```
class TXMLDisplayReferenceButton : public TWebButton;
```

File

MidItems

Description

Use TXMLDisplayReferenceButton as a base class for components that generate the HTML statements defining a specialized button control for working with a data display control in the Web page produced by TInetXPageProducer (see page 2285).

TXMLDisplayReferenceButton defines properties and methods for

Generating the HTML code to define the button.

Associating the button with another Web item that generates a data display control.

Specifying tag attributes of the HTML button control, including those defined by a style sheet.

Fitting the component into a hierarchy of Web items, each of which generates part of a larger HTML document.

In order to allow other Web items to use the HTML generated by the button control, TXMLDisplayReferenceButton inherits support for the IWebContent interface from TWebButton (see page 2058).

Do not create instances of TXMLDisplayReferenceButton. TXMLDisplayReferenceButton relies on anabstract or, in C++ terminology, pure virtual method to generate the HTML statements defining a button control. Instead, create descendants of TXMLDisplayReferenceButton, such as TPostButton (see page 1947), TInsertButton (see page 1935), and so on.

See Also

TFirstButton (see page 1913)

TNextButton (see page 1943)

TPriorButton (see page 1949)

TLastButton (see page 1937)

TNextPageButton (see page 1945)

TPriorPageButton (see page 1951)

TInsertButton (see page 1935)

TDeleteButton (see page 1867)

TUndoButton (see page 2055)

TPostButton (see page 1947)

IWebContent

TDataNavigator (see page 1855)

TResetQueryButton (see page 2022)

TSubmitQueryButton (see page 2038)

37.1.68.1 TXMLDisplayReferenceButton Members

The following tables list the members exposed by TXMLDisplayReferenceButton.

Protected Methods

	Name	Description
	Content (see page 2060)	Returns the HTML that defines the button control.
V	GetCaption (see page 2060)	Returns the value of the Caption (see page 2062) property.
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
V	GetLayoutAttributes (see page 2060)	Returns the layout object that specifies how the generated control should look.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
A	ImplContent (see page 2061)	Provides the underlying implementation for the Content (see page 2060) method.
V	Notification (see page 2194)	Forwards notification messages to all owned components.
V	SetCaption (see page 2061)	Sets the value of the Caption (see page 2062) property.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

Protected Properties

	Name	Description
R	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Methods

	Name	Description
V	Create (see page 2193)	Creates and initializes an instance of TXMLDisplayReferenceButton (see page 2191).
V	Destroy (see page 2193)	Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Public Properties

	Name	Description
	Caption (see page 2062)	Specifies the caption that appears on the button.
	Custom (see page 2062)	Represents a tag option and its value in the generated INPUT tag.
	Style (see page 2063)	Adds style attributes to the generated button control.
	StyleRule (see page 2063)	Applies a style in the page producer's style sheet to the button.
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

37.1.68.2 TXMLDisplayReferenceButton.Create Constructor

Creates and initializes an instance of TXMLDisplayReferenceButton (see page 2191).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TXMLDisplayReferenceButton(TComponent * AOwner);
```

Description

Call Create to instantiate a dataset button at runtime. Buttons added at design time using the Web Page editor are instantiated automatically.

37.1.68.3 TXMLDisplayReferenceButton.Destroy Destructor

Frees an instance of TXMLDisplayReferenceButton (see page 2191).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TXMLDisplayReferenceButton();
```

Description

Destroy removes the button from its parent's list of components that generate HTML.

37.1.68.4 TXMLDisplayReferenceButton Methods

The methods of the TXMLDisplayReferenceButton class are listed here.

Protected Methods

	Name	Description
	GetDisplayComponent (see page 2194)	This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	GetDisplayComponentParent (see page 2194)	This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.
	GetXmlDisplayName (see page 2194)	This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.
	Notification (see page 2194)	Forwards notification messages to all owned components.
	SetDisplayComponent (see page 2195)	This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.
	SetDisplayComponentParent (see page 2195)	This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

37.1.68.4.1 TXMLDisplayReferenceButton.GetDisplayComponent Method

Pascal

```
function GetDisplayComponent: TComponent;
```

C++

```
__fastcall TComponent * GetDisplayComponent();
```

Description

This is GetDisplayComponent, a member of class TXMLDisplayReferenceButton.

37.1.68.4.2 TXMLDisplayReferenceButton.GetDisplayComponentParent Method

Pascal

```
function GetDisplayComponentParent: Boolean;
```

C++

```
__fastcall Boolean GetDisplayComponentParent();
```

Description

This is GetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

37.1.68.4.3 TXMLDisplayReferenceButton.GetXmlDisplayName Method

Pascal

```
function GetXmlDisplayName: string;
```

C++

```
__fastcall UnicodeString GetXmlDisplayName();
```

Description

This is GetXmlDisplayName, a member of class TXMLDisplayReferenceButton.

37.1.68.4.4 TXMLDisplayReferenceButton.Notification Method

Forwards notification messages to all owned components.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Do not call the Notification method in an application. Notification is called automatically when the component specified by AComponent is about to be inserted or removed, as specified by Operation. By default, components pass along the notification to their owned components, if any.

A component can, if needed, act on the notification that a component is being inserted or removed. For example, if a component has object fields or properties that contain references to other components, it can check the notifications of component removals and invalidate those references as needed.

Note: Notification is not called for components that are freed implicitly (because their Owner is freed).

See Also

FreeNotification

InsertComponent

RemoveComponent

Owner

37.1.68.4.5 TXMLDisplayReferenceButton.SetDisplayComponent Method

Pascal

```
procedure SetDisplayComponent(const Value: TComponent);
```

C++

```
__fastcall SetDisplayComponent(const TComponent * Value);
```

Description

This is SetDisplayComponent, a member of class TXMLDisplayReferenceButton.

37.1.68.4.6 TXMLDisplayReferenceButton.SetDisplayComponentParent Method

Pascal

```
procedure SetDisplayComponentParent(const Value: Boolean);
```

C++

```
__fastcall SetDisplayComponentParent(const Boolean Value);
```

Description

This is SetDisplayComponentParent, a member of class TXMLDisplayReferenceButton.

37.1.68.5 TXMLDisplayReferenceButton Properties

The properties of the TXMLDisplayReferenceButton class are listed here.

Protected Properties

	Name	Description
	XMLDisplay (see page 2196)	This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

Public Properties

	Name	Description
	XMLComponent (see page 2195)	Associates the button with the component that generates the display control it influences.
	XMLUseParent (see page 2196)	Specifies whether the button is associated with its parent's display object.

37.1.68.5.1 TXMLDisplayReferenceButton.XMLComponent Property

Associates the button with the component that generates the display control it influences.

Pascal

```
property XMLComponent: TComponent;
```

C++

```
__property TComponent * XMLComponent;
```

Description

Set XMLComponent to specify the Web item that generates a display control that the button influences.

Note: Typically, descendants of TXMLDisplayReferenceButton (see page 2191) are used by a descendant of TXMLDisplayReferenceGroup (see page 2197), which has its own XMLComponent property. If the button influences this same component, you need not set the XMLComponent property. Instead, setting the XMLUseParent (see page 2196) property to true (the default) sets XMLComponent to the value of the parent's XMLComponent property.

See Also

XMLUseParent (see page 2196)

37.1.68.5.2 TXMLDisplayReferenceButton.XMLDisplay Property

Pascal

```
property XMLDisplay: TXMLDisplayParent;
```

C++

```
__property TXMLDisplayParent XMLDisplay;
```

Description

This is XMLDisplay, a member of class TXMLDisplayReferenceButton.

37.1.68.5.3 TXMLDisplayReferenceButton.XMLUseParent Property

Specifies whether the button is associated with its parent's display object.

Pascal

```
property XMLUseParent: Boolean;
```

C++

```
__property Boolean XMLUseParent;
```

Description

Use XMLUseParent to indicate how the dataset button determines the component that it influences. When XMLUseParent is false, the XMLComponent (see page 2195) property must be set to indicate the display component. When XMLUseParent is true, the button influences the component specified by its parent's XMLComponent (see page 2195) property.

The button's parent is the Web item that uses the button to assemble its own HTML content (typically a TDataNavigator (see page 1855) or TQueryButtons (see page 1962) object). If the parent does not have an associated display component, setting XMLUseParent to true sets the XMLComponent (see page 2195) property to nil (Delphi) or NULL (C++).

See Also

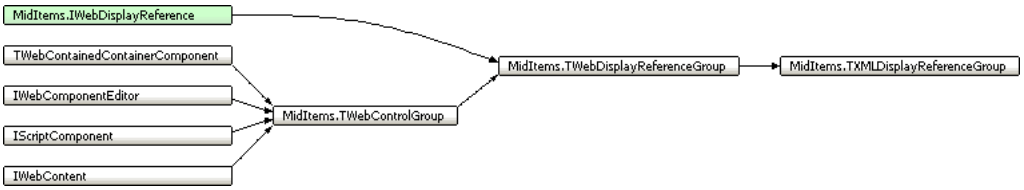
XMLComponent (see page 2195)

XMLComponent (see page 2199)

37.1.69 TXMLDisplayReferenceGroup Class

TXMLDisplayReferenceGroup is the base class for Web items that arrange a set of button controls that influence a control that displays data.

Class Hierarchy



Pascal

```
TXMLDisplayReferenceGroup = class(TWebDisplayReferenceGroup);
```

C++

```
class TXMLDisplayReferenceGroup : public TWebDisplayReferenceGroup;
```

File

MidItems

Description

Use TXMLDisplayReferenceGroup as a base class for Web items that generate the HTML to arrange a collection of button controls. Each TXMLDisplayGroup (see page 2176) object has an associated Web item that generates controls that display data in the final HTML document. The buttons arranged by TXMLDisplayReferenceGroup manipulate the information the display controls show.

In order to let other Web items use the HTML that the TXMLDisplayReferenceGroup object generates, TXMLDisplayReferenceGroup inherits the IWebContent interface from TWebControlGroup (see page 2064).

Do not create instances of TXMLDisplayReferenceGroup. TXMLDisplayReferenceGroup manages the Web items that generate buttons and the association to a display generator, but does not produce any content. Instead, use a descendant of TXMLDisplayReferenceGroup that generates the HTML for its set of specialized buttons such as TDataNavigator (see page 1855) or TQueryButtons (see page 1962).

See Also

TDataNavigator (see page 1855)

IWebContent

37.1.69.1 TXMLDisplayReferenceGroup Members

The following tables list the members exposed by TXMLDisplayReferenceGroup.

Protected Methods

	Name	Description
	AddElements (see page 2066)	Used internally to generate any script declarations needed by the group of controls.
	CanAddClass (see page 2067)	Specifies whether a Web item of a specified type can be added to the WebFieldControls (see page 2072) property list.

	ChangeFieldControls (see page 2067)	Responds when the group of HTML controls changes.
	Content (see page 2068)	Returns the HTML that defines the group of HTML controls.
V	GetDefaultWebComponents (see page 2068)	Returns the value of the DefaultWebComponents (see page 2072) property.
V	GetLayoutAttributes (see page 2069)	Returns an object that specifies how the generated group of controls should look.
	GetSubComponents (see page 2069)	Returns the value of the WebFieldControls (see page 2072) property.
	GetWebDisplayComponent (see page 2100)	This is GetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.
V	ImplAddElements (see page 2198)	Provides the underlying implementation for the AddElements method.
V	ImplCanAddClass (see page 2070)	Provides the underlying implementation for the CanAddClass (see page 2067) method.
V	ImplContent (see page 2070)	Provides the underlying implementation for the Content (see page 2068) method.
V	ImplGetSubComponents (see page 2071)	Provides the underlying implementation for the GetSubComponents (see page 2069) method.
V	Notification (see page 2101)	Forwards notification messages to all owned components.
V	SetDefaultWebComponents (see page 2071)	Sets the value of the DefaultWebComponents (see page 2072) property.
V	SetWebDisplayComponent (see page 2101)	This is SetWebDisplayComponent, a member of class TWebDisplayReferenceGroup.

Protected Properties

	Name	Description
	DefaultWebComponents (see page 2072)	Specifies the default value of the WebFieldControls (see page 2072) property when that property is not explicitly set.
R	WebDisplay (see page 2102)	This is WebDisplay, a member of class TWebDisplayReferenceGroup.
	WebDisplayComponent (see page 2102)	

Public Methods

	Name	Description
V	Create (see page 2065)	Creates and initializes an instance of TWebControlGroup (see page 2064).
V	Create (see page 2100)	Creates an instance of TWebDisplayReferenceGroup (see page 2098).
V	Destroy (see page 2066)	Frees an instance of TWebControlGroup (see page 2064).
V	Destroy (see page 2100)	Frees an instance of TWebDisplayReferenceGroup (see page 2098).
	GetWebDisplayComponent (see page 2228)	This is GetWebDisplayComponent, a member of class IWebDisplayReference.

Public Properties

	Name	Description
R	WebDisplayComponent (see page 2228)	
R	WebFieldControls (see page 2072)	Lists the components that generate HTML controls for the Web control group.
	XMLComponent (see page 2199)	Associates the group with a component that generates HTML display controls.

37.1.69.2 TXMLDisplayReferenceGroup Methods

The methods of the TXMLDisplayReferenceGroup class are listed here.

Protected Methods

	Name	Description
V	ImplAddElements (see page 2198)	Provides the underlying implementation for the AddElements method.

37.1.69.2.1 TXMLDisplayReferenceGroup.ImplAddElements Method

Provides the underlying implementation for the AddElements method.

Pascal

```
procedure ImplAddElements(AddIntf: IAddScriptElements); override;
```

C++

```
virtual __fastcall ImplAddElements(IAddScriptElements AddIntf);
```

Description

The AddElements method calls ImplAddElements to add any script declarations by calling the AddIntf interface. These declarations allow the generated display control to call script methods in response to user events by ensuring that the required methods are declared.

As implemented in TWebControlGroup, ImplAddElements does nothing. TWebControlGroup descendants override ImplAddElements to add any required declarations.

See Also

AddElements (see page 2066)

37.1.69.3 TXMLDisplayReferenceGroup Properties

The properties of the TXMLDisplayReferenceGroup class are listed here.

Public Properties

	Name	Description
	XMLComponent (see page 2199)	Associates the group with a component that generates HTML display controls.

37.1.69.3.1 TXMLDisplayReferenceGroup.XMLComponent Property

Associates the group with a component that generates HTML display controls.

Pascal

```
property XMLComponent: TComponent;
```

C++

```
__property TComponent * XMLComponent;
```

Description

Use XMLComponent to associate the group of buttons with a Web item that generates the display controls those buttons influence. Typical values of XMLComponent are TFieldGroup (see page 1869) or TDataGrid (see page 1850).

If all the buttons arranged by the TXMLDisplayReferenceGroup (see page 2197) object operate on the same set of display controls, it is most efficient to specify the Web item generating these controls as the value of XMLComponent. Then, each button uses this value to determine its associated display controls when its XMLUseParent property is true. Any buttons that do not operate on the display controls generated by XMLComponent can set their own XMLComponent property to another Web item.

See Also

XMLComponent (see page 2195)

XMLUseParent (see page 2196)

TFieldGroup (see page 1869)

TDataGrid (see page 1850)

TQueryFieldGroup (see page 1965)

37.2 Interfaces

The following table lists interfaces in this documentation.

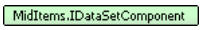
Interfaces

	Name	Description
	IDataSetComponent (see page 2200)	IDataSetComponent defines the methods that bind a Web item to an XML broker.
	IDataSetField (see page 2204)	This is class MidItems.IDataSetField.
	IDataSetFields (see page 2206)	This is class MidItems.IDataSetFields.
	IFormatColumn (see page 2208)	This is class MidItems.IFormatColumn.
	IHTMLField (see page 2209)	IHTMLField is an interface that must be implemented by Web items that generate <INPUT> elements.
	IHTMLForm (see page 2211)	IHTMLForm is an interface that must be implemented by Web items that generate forms.
	IInetXWebDataSet (see page 2213)	IInetXWebDataSet lets a Web item interact with information about a dataset in an XML broker's data packet.
	IQueryField (see page 2217)	This is class MidItems.IQueryField.
	IQueryFields (see page 2220)	This is class MidItems.IQueryFields.
	IQueryForm (see page 2221)	This is class MidItems.IQueryForm.
	IRestoreDefaults (see page 2221)	IRestoreDefaults enables the Web Page editor to restore a Web item to a default state.
	IStatusField (see page 2223)	This is class MidItems.IStatusField.
	IValidateField (see page 2223)	This is class MidItems.IValidateField.
	IValidateFields (see page 2224)	This is class MidItems.IValidateFields.
	IWebDisplay (see page 2226)	
	IWebDisplayReference (see page 2227)	
	IXMLDisplay (see page 2229)	This is class MidItems.IXMLDisplay.

37.2.1 IDataSetComponent Interface

IDataSetComponent defines the methods that bind a Web item to an XML broker.

Class Hierarchy



Pascal

```
IDataSetComponent = interface;
```

C++

```
__interface IDataSetComponent;
```

File

MidItems

Description

The TXMLDisplayGroup (see page 2176) class implements IDataSetComponent to provide information about the dataset from which its child controls obtain their data. The child Web items use this information to determine the default data binding.

Most Web items that generate a group of data-aware controls inherit the support for IDataSetComponent from TXMLDisplayGroup (see page 2176).

See Also

TXMLDisplayGroup (see page 2176)

TXMLBroker

37.2.1.1 IDataSetComponent Members

The following tables list the members exposed by IDataSetComponent.

Public Methods

	Name	Description
	GetDataSet (see page 2201)	Returns the value of the DataSet (see page 2202) property.
	GetXMLBroker (see page 2201)	Returns the value of the XMLBroker (see page 2203) property.
	GetXMLDataSet (see page 2202)	Returns the value of the XMLDataSet (see page 2203) property.

Public Properties

	Name	Description
R	DataSet (see page 2202)	Provides access to an interface for working with the associated dataset.
R	XMLBroker (see page 2203)	Indicates the XML broker that fetches the data packets to which the Web item is bound.
R	XMLDataSet (see page 2203)	Indicates the TXMLDataSet (see page 2164) object that represents the dataset to which the Web item is bound.

37.2.1.2 IDataSetComponent Methods

The methods of the IDataSetComponent class are listed here.

Public Methods

	Name	Description
	GetDataSet (see page 2201)	Returns the value of the DataSet (see page 2202) property.
	GetXMLBroker (see page 2201)	Returns the value of the XMLBroker (see page 2203) property.
	GetXMLDataSet (see page 2202)	Returns the value of the XMLDataSet (see page 2203) property.

37.2.1.2.1 IDataSetComponent.GetDataSet Method

Returns the value of the DataSet (see page 2202) property.

Pascal

```
function GetDataSet: IInetXWebDataSet;
```

C++

```
__fastcall IInetXWebDataSet GetDataSet();
```

Description

GetDataSet is the read access method for the DataSet (see page 2202) property.

See Also

DataSet (see page 2202)

37.2.1.2.2 IDataSetComponent.GetXMLBroker Method

Returns the value of the XMLBroker (see page 2203) property.

Pascal

```
function GetXMLBroker: TXMLBroker;
```

C++

```
__fastcall TXMLBroker GetXMLBroker();
```

Description

GetXMLBroker is the read access method for the XMLBroker (see page 2203) property.

See Also

XMLBroker (see page 2203)

37.2.1.2.3 IDataSetComponent.GetXMLDataSet Method

Returns the value of the XMLDataSet (see page 2203) property.

Pascal

```
function GetXMLDataSet: TXMLDataSet;
```

C++

```
__fastcall TXMLDataSet GetXMLDataSet();
```

Description

GetXMLDataSet is the read access method for the XMLDataSet (see page 2203) property.

See Also

XMLDataSet (see page 2203)

37.2.1.3 IDataSetComponent Properties

The properties of the IDataSetComponent class are listed here.

Public Properties

	Name	Description
R	DataSet (see page 2202)	Provides access to an interface for working with the associated dataset.
R	XMLBroker (see page 2203)	Indicates the XML broker that fetches the data packets to which the Web item is bound.
R	XMLDataSet (see page 2203)	Indicates the TXMLDataSet (see page 2164) object that represents the dataset to which the Web item is bound.

37.2.1.3.1 IDataSetComponent.DataSet Property

Provides access to an interface for working with the associated dataset.

Pascal

```
property DataSet: IInetXWebDataSet;
```

C++

```
__property IInetXWebDataSet DataSet;
```

Description

The child Web items of the item that implements IDataSetComponent (see page 2200) use the DataSet property to obtain information about the dataset to which they are bound and to respond when this information changes. This information includes

details about the available fields and parameters in the dataset to which they can be bound.

See Also

XMLDataSet (see page 2203)

37.2.1.3.2 IDataSetComponent.XMLBroker Property

Indicates the XML broker that fetches the data packets to which the Web item is bound.

Pascal

```
property XMLBroker: TXMLBroker;
```

C++

```
__property TXMLBroker XMLBroker;
```

Description

Use XMLBroker to access the TXMLBroker object that fetches XML data packets from the application server. The XML broker can be used to obtain information about the source of the dataset, the form that contains the XML data packet, the javascript variables that represent this data packet, and so on.

Note: The dataset represented by the Web item may not correspond directly to the XML broker's dataset. It may be a nested dataset contained by that dataset. The XMLDataSet (see page 2203) property indicates whether the dataset is nested, and if so, describes the field that represents it.

See Also

XMLDataSet (see page 2203)

DataSet (see page 2202)

37.2.1.3.3 IDataSetComponent.XMLDataSet Property

Indicates the TXMLDataSet (see page 2164) object that represents the dataset to which the Web item is bound.

Pascal

```
property XMLDataSet: TXMLDataSet;
```

C++

```
__property TXMLDataSet XMLDataSet;
```

Description

XMLDataSet describes how to locate the appropriate dataset, given the XML broker that provides its XML data packet. This dataset may be the main dataset of the XML broker, or it may be a nested dataset that is represented by a dataset field within that dataset (or a dataset field nested in a dataset field, and so on).

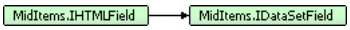
See Also

DataSet (see page 2202)

XMLBroker (see page 2203)

37.2.2 IDataSetField Interface

Class Hierarchy



Pascal

```
IDataSetField = interface(IHTMLField);
```

C++

```
__interface IDataSetField : public IHTMLField;
```

File

MidItems

Description

This is class MidItems.IDataSetField.

37.2.2.1 IDataSetField Members

The following tables list the members exposed by IDataSetField.

Public Methods

	Name	Description
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.

Public Properties

	Name	Description
	FieldName (see page 2205)	This is FieldName, a member of class IDataSetField.
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

37.2.2.2 IDataSetField Methods

The methods of the IDataSetField class are listed here.

Public Methods

	Name	Description
	GetFieldName (see page 2204)	This is GetFieldName, a member of class IDataSetField.
	GetRowSetFieldAttributes (see page 2205)	This is GetRowSetFieldAttributes, a member of class IDataSetField.
	SetFieldName (see page 2205)	This is SetFieldName, a member of class IDataSetField.

37.2.2.2.1 IDataSetField.GetFieldName Method

Pascal

```
function GetFieldName: string;
```

C++

```
__fastcall UnicodeString GetFieldName();
```

Description

This is GetFieldName, a member of class IDataSetField.

37.2.2.2.2 IDataSetField.GetRowSetFieldAttributes Method

Pascal

```
function GetRowSetFieldAttributes(const FieldVarName: string): string;
```

C++

```
__fastcall UnicodeString GetRowSetFieldAttributes(const UnicodeString FieldVarName);
```

Description

This is GetRowSetFieldAttributes, a member of class IDataSetField.

37.2.2.2.3 IDataSetField.SetFieldName Method

Pascal

```
procedure SetFieldName(AFieldName: string);
```

C++

```
__fastcall SetFieldName(UnicodeString AFieldName);
```

Description

This is SetFieldName, a member of class IDataSetField.

37.2.2.3 IDataSetField Properties

The properties of the IDataSetField class are listed here.

Public Properties

	Name	Description
	FieldName (see page 2205)	This is FieldName, a member of class IDataSetField.

37.2.2.3.1 IDataSetField.FieldName Property

Pascal

```
property FieldName: string;
```

C++

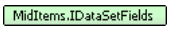
```
__property UnicodeString FieldName;
```

Description

This is FieldName, a member of class IDataSetField.

37.2.3 IDataSetFields Interface

Class Hierarchy



Pascal

```
IDataSetFields = interface;
```

C++

```
__interface IDataSetFields;
```

File

MidItems

Description

This is class MidItems.IDataSetFields.

37.2.3.1 IDataSetFields Members

The following tables list the members exposed by IDataSetFields.

Public Methods

	Name	Description
	GetFieldsList (see page 2207)	This is GetFieldsList, a member of class IDataSetFields.
	GetVisibleFields (see page 2207)	This is GetVisibleFields, a member of class IDataSetFields.
	HasStatusField (see page 2207)	This is HasStatusField, a member of class IDataSetFields.
	IsFieldInUse (see page 2207)	This is IsFieldInUse, a member of class IDataSetFields.
	ValidateDataSet (see page 2207)	This is ValidateDataSet, a member of class IDataSetFields.

Public Properties

	Name	Description
	VisibleFields (see page 2208)	This is VisibleFields, a member of class IDataSetFields.

37.2.3.2 IDataSetFields Methods

The methods of the IDataSetFields class are listed here.

Public Methods

	Name	Description
	GetFieldsList (see page 2207)	This is GetFieldsList, a member of class IDataSetFields.
	GetVisibleFields (see page 2207)	This is GetVisibleFields, a member of class IDataSetFields.
	HasStatusField (see page 2207)	This is HasStatusField, a member of class IDataSetFields.
	IsFieldInUse (see page 2207)	This is IsFieldInUse, a member of class IDataSetFields.
	ValidateDataSet (see page 2207)	This is ValidateDataSet, a member of class IDataSetFields.

37.2.3.2.1 IDataSetFields.GetFieldsList Method

Pascal

```
procedure GetFieldsList(AList: TStrings);
```

C++

```
__fastcall GetFieldsList(TStrings AList);
```

Description

This is GetFieldsList, a member of class IDataSetFields.

37.2.3.2.2 IDataSetFields.GetVisibleFields Method

Pascal

```
function GetVisibleFields: TWebComponentList;
```

C++

```
__fastcall TWebComponentList GetVisibleFields();
```

Description

This is GetVisibleFields, a member of class IDataSetFields.

37.2.3.2.3 IDataSetFields.HasStatusField Method

Pascal

```
function HasStatusField: Boolean;
```

C++

```
__fastcall Boolean HasStatusField();
```

Description

This is HasStatusField, a member of class IDataSetFields.

37.2.3.2.4 IDataSetFields.IsFieldInUse Method

Pascal

```
function IsFieldInUse(AName: string): Boolean;
```

C++

```
__fastcall Boolean IsFieldInUse(UnicodeString AName);
```

Description

This is IsFieldInUse, a member of class IDataSetFields.

37.2.3.2.5 IDataSetFields.ValidateDataSet Method

Pascal

```
procedure ValidateDataSet;
```

C++

```
__fastcall ValidateDataSet();
```

Description

This is ValidateDataSet, a member of class IDataSetFields.

37.2.3.3 IDataSetFields Properties

The properties of the IDataSetFields class are listed here.

Public Properties

	Name	Description
	VisibleFields (see page 2208)	This is VisibleFields, a member of class IDataSetFields.

37.2.3.3.1 IDataSetFields.VisibleFields Property

Pascal

```
property VisibleFields: TWebComponentList;
```

C++

```
__property TWebComponentList VisibleFields;
```

Description

This is VisibleFields, a member of class IDataSetFields.

37.2.4 IFormatColumn Interface

Class Hierarchy

MidItems.IFormatColumn

Pascal

```
IFormatColumn = interface;
```

C++

```
__interface IFormatColumn;
```

File

MidItems

Description

This is class MidItems.IFormatColumn.

37.2.4.1 IFormatColumn Members

The following tables list the members exposed by IFormatColumn.

Public Methods

	Name	Description
	FormatColumnData (see page 2209)	This is FormatColumnData, a member of class IFormatColumn.
	FormatColumnHeading (see page 2209)	This is FormatColumnHeading, a member of class IFormatColumn.

37.2.4.2 IFormatColumn Methods

The methods of the IFormatColumn class are listed here.

Public Methods

	Name	Description
	FormatColumnData (see page 2209)	This is FormatColumnData, a member of class IFormatColumn.
	FormatColumnHeading (see page 2209)	This is FormatColumnHeading, a member of class IFormatColumn.

37.2.4.2.1 IFormatColumn.FormatColumnData Method

Pascal

```
function FormatColumnData(const Content: string; Options: TWebContentOptions): string;
```

C++

```
__fastcall UnicodeString FormatColumnData(const UnicodeString Content, TWebContentOptions
Options);
```

Description

This is FormatColumnData, a member of class IFormatColumn.

37.2.4.2.2 IFormatColumn.FormatColumnHeading Method

Pascal

```
function FormatColumnHeading(Options: TWebContentOptions): string;
```

C++

```
__fastcall UnicodeString FormatColumnHeading(TWebContentOptions Options);
```

Description

This is FormatColumnHeading, a member of class IFormatColumn.

37.2.5 IHTMLField Interface

IHTMLField is an interface that must be implemented by Web items that generate <INPUT> elements.

Class Hierarchy

MidItems.IHTMLField

Pascal

```
IHTMLField = interface;
```

C++

```
__interface IHTMLField;
```

File

MidItems

Description

The TWebDataDisplay (see page 2073) class implements IHTMLField to provide information about information about the HTML control it represents. This interface is used when generating javascript that refers to the control.

Most Web items that generate HTML <INPUT> elements inherit the support for IHTMLField from TWebDataDisplay (see page 2073). However, when writing custom Web items that generate <INPUT> tags but that do not descend from TWebDataDisplay (see page 2073), component writers must ensure that the Web item supports the IHTMLField interface.

See Also

TWebDataDisplay (see page 2073)

37.2.5.1 IHTMLField Members

The following tables list the members exposed by IHTMLField.

Public Methods

	Name	Description
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.

Public Properties

	Name	Description
	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

37.2.5.2 IHTMLField Methods

The methods of the IHTMLField class are listed here.

Public Methods

	Name	Description
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.

37.2.5.2.1 IHTMLField.GetHTMLControlName Method

Returns the name of the generated HTML <INPUT> control.

Pascal

```
function GetHTMLControlName: string;
```

C++

```
__fastcall UnicodeString GetHTMLControlName();
```

Description

GetHTMLFormName is the read access method for the HTMLControlName (see page 2211) property.

See Also

HTMLControlName (see page 2211)

37.2.5.3 IHTMLField Properties

The properties of the IHTMLField class are listed here.

Public Properties

	Name	Description
	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.

37.2.5.3.1 IHTMLField.HTMLControlName Property

Indicates the name of the generated HTML control.

Pascal

```
property HTMLControlName: string;
```

C++

```
__property UnicodeString HTMLControlName;
```

Description

HTMLControlName is the value of the NAME option on the generated <INPUT> tag. This value is used when creating javascript references to the control.

37.2.6 IHTMLForm Interface

IHTMLForm is an interface that must be implemented by Web items that generate forms.

Class Hierarchy

MidItems.IHTMLForm

Pascal

```
IHTMLForm = interface;
```

C++

```
__interface IHTMLForm;
```

File

MidItems

Description

Web items that produce forms (such as TQueryForm (see page 1971) or TDataForm (see page 1847)) must implement the IHTML form interface. This interface allows other Web items to access information about form elements when generating javascript declarations and method calls.

See Also

TQueryForm (see page 1971)

TDataForm (see page 1847)

37.2.6.1 IHTMLForm Members

The following tables list the members exposed by IHTMLForm.

Public Methods

	Name	Description
	GetHTMLFormName (see page 2212)	Returns the name of the generated HTML form.
	GetHTMLFormVarName (see page 2212)	Returns the name of the javascript variable that refers to the generated form.

Public Properties

	Name	Description
R	HTMLFormName (see page 2213)	Indicates the name of the generated HTML form.
R	HTMLFormVarName (see page 2213)	Indicates the name of the javascript variable that represents the HTML form.

37.2.6.2 IHTMLForm Methods

The methods of the IHTMLForm class are listed here.

Public Methods

	Name	Description
	GetHTMLFormName (see page 2212)	Returns the name of the generated HTML form.
	GetHTMLFormVarName (see page 2212)	Returns the name of the javascript variable that refers to the generated form.

37.2.6.2.1 IHTMLForm.GetHTMLFormName Method

Returns the name of the generated HTML form.

Pascal

```
function GetHTMLFormName: string;
```

C++

```
__fastcall UnicodeString GetHTMLFormName();
```

Description

GetHTMLFormName is the read access method for the HTMLFormName (see page 2213) property.

See Also

HTMLFormName (see page 2213)

37.2.6.2.2 IHTMLForm.GetHTMLFormVarName Method

Returns the name of the javascript variable that refers to the generated form.

Pascal

```
function GetHTMLFormVarName: string;
```

C++

```
__fastcall UnicodeString GetHTMLFormVarName();
```

Description

GetHTMLFormVarName is the read access method for the HTMLFormVarName (see page 2213) property.

See Also

HTMLFormVarName (see page 2213)

37.2.6.3 IHTMLForm Properties

The properties of the IHTMLForm class are listed here.

Public Properties

	Name	Description
R	HTMLFormName (see page 2213)	Indicates the name of the generated HTML form.
R	HTMLFormVarName (see page 2213)	Indicates the name of the javascript variable that represents the HTML form.

37.2.6.3.1 IHTMLForm.HTMLFormName Property

Indicates the name of the generated HTML form.

Pascal

```
property HTMLFormName: string;
```

C++

```
__property UnicodeString HTMLFormName;
```

Description

HTMLFormName is the value of the NAME option on the generated form. Other Web items can use this property when they need to refer to the form name. For example, this property is used to declare the javascript variable that represents the form.

See Also

HTMLFormVarName (see page 2213)

37.2.6.3.2 IHTMLForm.HTMLFormVarName Property

Indicates the name of the javascript variable that represents the HTML form.

Pascal

```
property HTMLFormVarName: string;
```

C++

```
__property UnicodeString HTMLFormVarName;
```

Description

HTMLFormVarName is the value of the javascript variable for the form. Other Web items can use this property when they need to refer to the form in generated javascript code.

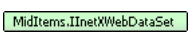
See Also

HTMLFormName (see page 2213)

37.2.7 IInetXWebDataSet Interface

IInetXWebDataSet lets a Web item interact with information about a dataset in an XML broker's data packet.

Class Hierarchy



Pascal

```
IInetXWebDataSet = interface;
```

C++

```
__interface IInetXWebDataSet;
```

File

MidItems

Description

Web items use the IInetXWebDataSet interface (formerly known as IMidasWebDataSet (see page 2240)) to obtain information about a dataset supplied by an XML broker. By using this interface, the Web item does not need to worry about whether the dataset is the top-level dataset in the XML data packet or a nested dataset in that data packet.

IInetXWebDataSet defines methods for obtaining information about the fields and parameters of the associated dataset. It also defines methods for assigning an event handler that receives notifications when this information changes.

Note: VCL Method declarations that use IInetXWebDataSet use the _di_IInetXWebDataSet type instead. This type is a DelphiInterface wrapper around the IInetXWebDataSet interface:

```
typedef System::DelphiInterfaceDelphiInterface_object< IInetXWebDataSet >
_di_IInetXWebDataSet;
```

See Also

TXMLDataSet (see page 2164)

TXMLBroker

37.2.7.1 IInetXWebDataSet Members

The following tables list the members exposed by IInetXWebDataSet.

Public Methods

	Name	Description
	AddOnDataChange (see page 2215)	Adds an event handler to the list of event handlers that are called when the fields or parameters of the dataset change.
	FetchParams (see page 2215)	Tells the XML broker that manages the associated dataset to fetch parameter values from the application server.
	FieldCount (see page 2215)	Returns the number of fields in the associated dataset.
	Fields (see page 2216)	Returns the set of fields in the associated dataset.
	ParamCount (see page 2216)	Returns the number of parameters for the associated dataset.
	Params (see page 2216)	Returns the set of parameters in the associated dataset.
	RemoveOnDataChange (see page 2217)	Removes an event handler from the list of event handlers that are called when the fields or parameters of the dataset change.

37.2.7.2 IInetXWebDataSet Methods

The methods of the IInetXWebDataSet class are listed here.

Public Methods

	Name	Description
	AddOnDataChange (see page 2215)	Adds an event handler to the list of event handlers that are called when the fields or parameters of the dataset change.

	FetchParams (see page 2215)	Tells the XML broker that manages the associated dataset to fetch parameter values from the application server.
	FieldCount (see page 2215)	Returns the number of fields in the associated dataset.
	Fields (see page 2216)	Returns the set of fields in the associated dataset.
	ParamCount (see page 2216)	Returns the number of parameters for the associated dataset.
	Params (see page 2216)	Returns the set of parameters in the associated dataset.
	RemoveOnDataChange (see page 2217)	Removes an event handler from the list of event handlers that are called when the fields or parameters of the dataset change.

37.2.7.2.1 IInetXWebDataSet.AddOnDataChange Method

Adds an event handler to the list of event handlers that are called when the fields or parameters of the dataset change.

Pascal

```
procedure AddOnDataChange(const Value: TNotifyEvent);
```

C++

```
__fastcall AddOnDataChange(const TNotifyEvent Value);
```

Description

Call AddOnDataChange to specify an event handler that is called when the fields or parameters of the dataset change. This allows Web items to update their data bindings to reflect changes in the associated data packet.

Value specifies the event handler that is added to the list.

See Also

RemoveOnDataChange (see page 2217)

37.2.7.2.2 IInetXWebDataSet.FetchParams Method

Tells the XML broker that manages the associated dataset to fetch parameter values from the application server.

Pascal

```
procedure FetchParams;
```

C++

```
__fastcall FetchParams();
```

Description

Call FetchParams to tell the XML broker to fetch parameter values from the application server. This ensures that the Params (see page 2216) method returns the current values of any parameters on the server.

See Also

FetchParams

Params (see page 2216)

37.2.7.2.3 IInetXWebDataSet.FieldCount Method

Returns the number of fields in the associated dataset.

Pascal

```
function FieldCount: Integer;
```

C++

```
__fastcall int FieldCount();
```

Description

Call FieldCount to determine the number of fields returned by the Fields (see page 2216) method. This is the number of fields in the XML data packet belonging to the associated dataset.

See Also

Fields (see page 2216)

37.2.7.2.4 IInetXWebDataSet.Fields Method

Returns the set of fields in the associated dataset.

Pascal

```
function Fields: TFields;
```

C++

```
__fastcall TFields Fields();
```

Description

Call Fields to obtain access to the field objects (TField) that represent the fields in the XML data packet. Each field object can be used to read the value of the field, determine its type, and so on.

See Also

FieldCount (see page 2215)

37.2.7.2.5 IInetXWebDataSet.ParamCount Method

Returns the number of parameters for the associated dataset.

Pascal

```
function ParamCount: Integer;
```

C++

```
__fastcall int ParamCount();
```

Description

Call ParamCount to determine the number of parameters returned by the Params (see page 2216) method. This is the number of parameters in the XML data packet belonging to the associated dataset.

See Also

Params (see page 2216)

37.2.7.2.6 IInetXWebDataSet.Params Method

Returns the set of parameters in the associated dataset.

Pascal

```
function Params: TParams;
```

C++

```
__fastcall TParams Params();
```

Description

Call Params to obtain access to the parameter objects (TParam) that represent the parameters in the XML data packet. Each parameter object can be used to read the value of the parameter, determine its type, and so on.

See Also

ParamCount (see page 2216)

FetchParams (see page 2215)

37.2.7.2.7 IInetXWebDataSet.RemoveOnDataChange Method

Removes an event handler from the list of event handlers that are called when the fields or parameters of the dataset change.

Pascal

```
procedure RemoveOnDataChange(const Value: TNotifyEvent);
```

C++

```
__fastcall RemoveOnDataChange(const TNotifyEvent Value);
```

Description

Call RemoveOnDataChange to remove an event handler that was added using the AddOnDataChange (see page 2215) method.

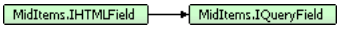
Value specifies the event handler to remove from the list.

See Also

AddOnDataChange (see page 2215)

37.2.8 IQueryField Interface

Class Hierarchy



Pascal

```
IQueryField = interface(IHTMLField);
```

C++

```
__interface IQueryField : public IHTMLField;
```

File

MidItems

Description

This is class MidItems.IQueryField.

37.2.8.1 IQueryField Members

The following tables list the members exposed by IQueryField.

Public Methods

	Name	Description
	GetHTMLControlName (see page 2210)	Returns the name of the generated HTML <INPUT> control.
	GetParamName (see page 2218)	This is GetParamName, a member of class IQueryField.
	GetText (see page 2218)	This is GetText, a member of class IQueryField.
	SetParamName (see page 2218)	This is SetParamName, a member of class IQueryField.
	SetText (see page 2219)	This is SetText, a member of class IQueryField.

Public Properties

	Name	Description
R	HTMLControlName (see page 2211)	Indicates the name of the generated HTML control.
	ParamName (see page 2219)	This is ParamName, a member of class IQueryField.
	Text (see page 2219)	This is Text, a member of class IQueryField.

37.2.8.2 IQueryField Methods

The methods of the IQueryField class are listed here.

Public Methods

	Name	Description
	GetParamName (see page 2218)	This is GetParamName, a member of class IQueryField.
	GetText (see page 2218)	This is GetText, a member of class IQueryField.
	SetParamName (see page 2218)	This is SetParamName, a member of class IQueryField.
	SetText (see page 2219)	This is SetText, a member of class IQueryField.

37.2.8.2.1 IQueryField.GetParamName Method

Pascal

```
function GetParamName: string;
```

C++

```
__fastcall UnicodeString GetParamName();
```

Description

This is GetParamName, a member of class IQueryField.

37.2.8.2.2 IQueryField.GetText Method

Pascal

```
function GetText: string;
```

C++

```
__fastcall UnicodeString GetText();
```

Description

This is GetText, a member of class IQueryField.

37.2.8.2.3 IQueryField.SetParamName Method

Pascal

```
procedure SetParamName(AParamName: string);
```

C++

```
__fastcall SetParamName(UnicodeString AParamName);
```

Description

This is SetParamName, a member of class IQueryField.

37.2.8.2.4 IQueryField.SetText Method

Pascal

```
procedure SetText(const Value: string);
```

C++

```
__fastcall SetText(const UnicodeString Value);
```

Description

This is SetText, a member of class IQueryField.

37.2.8.3 IQueryField Properties

The properties of the IQueryField class are listed here.

Public Properties

	Name	Description
	ParamName (see page 2219)	This is ParamName, a member of class IQueryField.
	Text (see page 2219)	This is Text, a member of class IQueryField.

37.2.8.3.1 IQueryField.ParamName Property

Pascal

```
property ParamName: string;
```

C++

```
__property UnicodeString ParamName;
```

Description

This is ParamName, a member of class IQueryField.

37.2.8.3.2 IQueryField.Text Property

Pascal

```
property Text: string;
```

C++

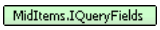
```
__property UnicodeString Text;
```

Description

This is Text, a member of class IQueryField.

37.2.9 IQueryFields Interface

Class Hierarchy



Pascal

```
IQueryFields = interface;
```

C++

```
__interface IQueryFields;
```

File

MidItems

Description

This is class MidItems.IQueryFields.

37.2.9.1 IQueryFields Members

The following tables list the members exposed by IQueryFields.

Public Methods

	Name	Description
	GetParamsList (see page 2220)	This is GetParamsList, a member of class IQueryFields.
	IsParamInUse (see page 2221)	This is IsParamInUse, a member of class IQueryFields.
	ValidateDataSet (see page 2221)	This is ValidateDataSet, a member of class IQueryFields.

37.2.9.2 IQueryFields Methods

The methods of the IQueryFields class are listed here.

Public Methods

	Name	Description
	GetParamsList (see page 2220)	This is GetParamsList, a member of class IQueryFields.
	IsParamInUse (see page 2221)	This is IsParamInUse, a member of class IQueryFields.
	ValidateDataSet (see page 2221)	This is ValidateDataSet, a member of class IQueryFields.

37.2.9.2.1 IQueryFields.GetParamsList Method

Pascal

```
procedure GetParamsList(AList: TStrings);
```

C++

```
__fastcall GetParamsList(TStrings AList);
```

Description

This is GetParamsList, a member of class IQueryFields.

37.2.9.2.2 IQueryFields.IsParamInUse Method

Pascal

```
function IsParamInUse(AName: string): Boolean;
```

C++

```
__fastcall Boolean IsParamInUse(UnicodeString AName);
```

Description

This is IsParamInUse, a member of class IQueryFields.

37.2.9.2.3 IQueryFields.ValidateDataSet Method

Pascal

```
procedure ValidateDataSet;
```

C++

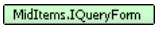
```
__fastcall ValidateDataSet();
```

Description

This is ValidateDataSet, a member of class IQueryFields.

37.2.10 IQueryForm Interface

Class Hierarchy



Pascal

```
IQueryForm = interface;
```

C++

```
__interface IQueryForm;
```

File

MidItems

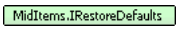
Description

This is class MidItems.IQueryForm.

37.2.11 IRestoreDefaults Interface

IRestoreDefaults enables the Web Page editor to restore a Web item to a default state.

Class Hierarchy



Pascal

```
IRestoreDefaults = interface;
```

C++

```
__interface IRestoreDefaults;
```

File

MidItems

Description

Implement IRestoreDefaults to allow the Web Page editor to restore a Web item to its default state. When a Web item implements IRestoreDefaults, the Web Page editor enables the Restore Defaults command when that item is selected.

IRestoreDefaults defines a single method, RestoreDefaults (see page 2222), which is called when a user chooses the Restore Defaults command in the Web Page editor.

See Also

IWebComponentEditor

37.2.11.1 IRestoreDefaults Members

The following tables list the members exposed by IRestoreDefaults.

Public Methods

	Name	Description
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.

37.2.11.2 IRestoreDefaults Methods

The methods of the IRestoreDefaults class are listed here.

Public Methods

	Name	Description
	RestoreDefaults (see page 2222)	Returns the Web item to a default state.

37.2.11.2.1 IRestoreDefaults.RestoreDefaults Method

Returns the Web item to a default state.

Pascal

```
procedure RestoreDefaults;
```

C++

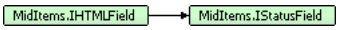
```
__fastcall RestoreDefaults();
```

Description

The Web page editor calls RestoreDefaults when the user chooses the Restore Defaults command. Web items implement this method to return property values to a default state.

37.2.12 IStatusField Interface

Class Hierarchy



Pascal

```
IStatusField = interface(IHTMLField);
```

C++

```
__interface IStatusField : public IHTMLField;
```

File

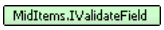
MidItems

Description

This is class MidItems.IStatusField.

37.2.13 IValidateField Interface

Class Hierarchy



Pascal

```
IValidateField = interface;
```

C++

```
__interface IValidateField;
```

File

MidItems

Description

This is class MidItems.IValidateField.

37.2.13.1 IValidateField Members

The following tables list the members exposed by IValidateField.

Public Methods

	Name	Description
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

37.2.13.2 IValidateField Methods

The methods of the IValidateField class are listed here.

Public Methods

	Name	Description
	ValidateField (see page 2224)	This is ValidateField, a member of class IValidateField.

37.2.13.2.1 IValidateField.ValidateField Method

Pascal

```
function ValidateField(DataSet: IInetXWebDataSet; AddIntf: IAddScriptElements): Boolean;
```

C++

```
__fastcall Boolean ValidateField(IInetXWebDataSet DataSet, IAddScriptElements AddIntf);
```

Description

This is ValidateField, a member of class IValidateField.

37.2.14 IValidateFields Interface

Class Hierarchy

MidItems.IValidateFields

Pascal

```
IValidateFields = interface;
```

C++

```
__interface IValidateFields;
```

File

MidItems

Description

This is class MidItems.IValidateFields.

37.2.14.1 IValidateFields Members

The following tables list the members exposed by IValidateFields.

Public Methods

	Name	Description
	GetEnableValidateFields (see page 2225)	This is GetEnableValidateFields, a member of class IValidateFields.
	SetEnableValidateFields (see page 2225)	This is SetEnableValidateFields, a member of class IValidateFields.
	ValidateFields (see page 2225)	This is ValidateFields, a member of class IValidateFields.

Public Properties

	Name	Description
	EnableValidateFields (see page 2226)	This is EnableValidateFields, a member of class IValidateFields.

37.2.14.2 IValidateFields Methods

The methods of the IValidateFields class are listed here.

Public Methods

	Name	Description
	GetEnableValidateFields (see page 2225)	This is GetEnableValidateFields, a member of class IValidateFields.
	SetEnableValidateFields (see page 2225)	This is SetEnableValidateFields, a member of class IValidateFields.
	ValidateFields (see page 2225)	This is ValidateFields, a member of class IValidateFields.

37.2.14.2.1 IValidateFields.GetEnableValidateFields Method

Pascal

```
function GetEnableValidateFields: Boolean;
```

C++

```
__fastcall Boolean GetEnableValidateFields();
```

Description

This is GetEnableValidateFields, a member of class IValidateFields.

37.2.14.2.2 IValidateFields.SetEnableValidateFields Method

Pascal

```
procedure SetEnableValidateFields(Value: Boolean);
```

C++

```
__fastcall SetEnableValidateFields(Boolean Value);
```

Description

This is SetEnableValidateFields, a member of class IValidateFields.

37.2.14.2.3 IValidateFields.ValidateFields Method

Pascal

```
function ValidateFields(AddIntf: IAddScriptElements): Boolean;
```

C++

```
__fastcall Boolean ValidateFields(IAddScriptElements AddIntf);
```

Description

This is ValidateFields, a member of class IValidateFields.

37.2.14.3 IValidateFields Properties

The properties of the IValidateFields class are listed here.

Public Properties

	Name	Description
	EnableValidateFields (see page 2226)	This is EnableValidateFields, a member of class IValidateFields.

37.2.14.3.1 IValidateFields.EnableValidateFields Property

Pascal

```
property EnableValidateFields: Boolean;
```

C++

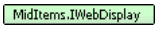
```
__property Boolean EnableValidateFields;
```

Description

This is EnableValidateFields, a member of class IValidateFields.

37.2.15 IWebDisplay Interface

Class Hierarchy



Pascal

```
IWebDisplay = interface;
```

C++

```
__interface IWebDisplay;
```

File

MidItems

See Also

TWebDisplay (see page 2095)

37.2.15.1 IWebDisplay Members

The following tables list the members exposed by IWebDisplay.

Public Methods

	Name	Description
	GetIsMultipleRecordView (see page 2227)	This is GetIsMultipleRecordView, a member of class IWebDisplay.

Public Properties

	Name	Description
R	IsMultipleRecordView (see page 2227)	

37.2.15.2 IWebDisplay Methods

The methods of the IWebDisplay class are listed here.

Public Methods

	Name	Description
	GetIsMultipleRecordView (see page 2227)	This is GetIsMultipleRecordView, a member of class IWebDisplay.

37.2.15.2.1 IWebDisplay.GetIsMultipleRecordView Method

Pascal

```
function GetIsMultipleRecordView: Boolean;
```

C++

```
__fastcall Boolean GetIsMultipleRecordView();
```

Description

This is GetIsMultipleRecordView, a member of class IWebDisplay.

37.2.15.3 IWebDisplay Properties

The properties of the IWebDisplay class are listed here.

Public Properties

	Name	Description
	IsMultipleRecordView (see page 2227)	

37.2.15.3.1 IWebDisplay.IsMultipleRecordView Property

Pascal

```
property IsMultipleRecordView: Boolean;
```

C++

```
__property Boolean IsMultipleRecordView;
```

Description

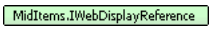
.

See Also

GetIsMultipleRecordView (see page 2227)

37.2.16 IWebDisplayReference Interface

Class Hierarchy



Pascal

```
IWebDisplayReference = interface;
```

C++

```
__interface IWebDisplayReference;
```

File

MidItems

See Also

TWebDisplay (see page 2095)

37.2.16.1 IWebDisplayReference Members

The following tables list the members exposed by IWebDisplayReference.

Public Methods

	Name	Description
	GetWebDisplayComponent (see page 2228)	This is GetWebDisplayComponent, a member of class IWebDisplayReference.

Public Properties

	Name	Description
R	WebDisplayComponent (see page 2228)	

37.2.16.2 IWebDisplayReference Methods

The methods of the IWebDisplayReference class are listed here.

Public Methods

	Name	Description
	GetWebDisplayComponent (see page 2228)	This is GetWebDisplayComponent, a member of class IWebDisplayReference.

37.2.16.2.1 IWebDisplayReference.GetWebDisplayComponent Method

Pascal

```
function GetWebDisplayComponent: TComponent;
```

C++

```
__fastcall TComponent * GetWebDisplayComponent();
```

Description

This is GetWebDisplayComponent, a member of class IWebDisplayReference.

37.2.16.3 IWebDisplayReference Properties

The properties of the IWebDisplayReference class are listed here.

Public Properties

	Name	Description
R	WebDisplayComponent (see page 2228)	

37.2.16.3.1 IWebDisplayReference.WebDisplayComponent Property

Pascal

```
property WebDisplayComponent: TComponent;
```

C++

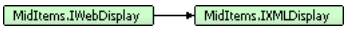
```
__property TComponent * WebDisplayComponent;
```

See Also

GetWebDisplayComponent (see page 2228)

37.2.17 IXMLDisplay Interface

Class Hierarchy



Pascal

```
IXMLDisplay = interface(IWebDisplay);
```

C++

```
__interface IXMLDisplay : public IWebDisplay;
```

File

MidItems

Description

This is class MidItems.IXMLDisplay.

37.2.17.1 IXMLDisplay Members

The following tables list the members exposed by IXMLDisplay.

Public Methods

	Name	Description
	GetIsMultipleRecordView (see page 2229)	This is GetIsMultipleRecordView, a member of class IXMLDisplay.
	GetXMLDisplayName (see page 2230)	This is GetXMLDisplayName, a member of class IXMLDisplay.
	GetXMLRowSetName (see page 2230)	This is GetXMLRowSetName, a member of class IXMLDisplay.

Public Properties

	Name	Description
R	IsMultipleRecordView (see page 2230)	
R	XMLDisplayName (see page 2231)	This is XMLDisplayName, a member of class IXMLDisplay.
R	XMLRowSetName (see page 2231)	This is XMLRowSetName, a member of class IXMLDisplay.

37.2.17.2 IXMLDisplay Methods

The methods of the IXMLDisplay class are listed here.

Public Methods

	Name	Description
	GetIsMultipleRecordView (see page 2229)	This is GetIsMultipleRecordView, a member of class IXMLDisplay.
	GetXMLDisplayName (see page 2230)	This is GetXMLDisplayName, a member of class IXMLDisplay.
	GetXMLRowSetName (see page 2230)	This is GetXMLRowSetName, a member of class IXMLDisplay.

37.2.17.2.1 IXMLDisplay.GetIsMultipleRecordView Method

Pascal

```
function GetIsMultipleRecordView: Boolean;
```

C++

```
__fastcall Boolean GetIsMultipleRecordView();
```

Description

This is GetIsMultipleRecordView, a member of class IXMLDisplay.

37.2.17.2.2 IXMLDisplay.GetXMLDisplayName Method

Pascal

```
function GetXMLDisplayName: string;
```

C++

```
__fastcall UnicodeString GetXMLDisplayName();
```

Description

This is GetXMLDisplayName, a member of class IXMLDisplay.

37.2.17.2.3 IXMLDisplay.GetXMLRowSetName Method

Pascal

```
function GetXMLRowSetName: string;
```

C++

```
__fastcall UnicodeString GetXMLRowSetName();
```

Description

This is GetXMLRowSetName, a member of class IXMLDisplay.

37.2.17.3 IXMLDisplay Properties

The properties of the IXMLDisplay class are listed here.

Public Properties

	Name	Description
R	IsMultipleRecordView (see page 2230)	
R	XMLDisplayName (see page 2231)	This is XMLDisplayName, a member of class IXMLDisplay.
R	XMLRowSetName (see page 2231)	This is XMLRowSetName, a member of class IXMLDisplay.

37.2.17.3.1 IXMLDisplay.IsMultipleRecordView Property

Pascal

```
property IsMultipleRecordView: Boolean;
```

C++

```
__property Boolean IsMultipleRecordView;
```

Description

.

See Also

GetIsMultipleRecordView (see page 2227)

37.2.17.3.2 IXMLDisplay.XMLDisplayName Property

Pascal

```
property XMLDisplayName: string;
```

C++

```
__property UnicodeString XMLDisplayName;
```

Description

This is XMLDisplayName, a member of class IXMLDisplay.

37.2.17.3.3 IXMLDisplay.XMLRowSetName Property

Pascal

```
property XMLRowSetName: string;
```

C++

```
__property UnicodeString XMLRowSetName;
```

Description

This is XMLRowSetName, a member of class IXMLDisplay.

37.3 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	AddBoolAttrib (see page 2232)	This is function MidItems.AddBoolAttrib.
	AddCustomAttrib (see page 2232)	This is function MidItems.AddCustomAttrib.
	AddIntAttrib (see page 2233)	This is function MidItems.AddIntAttrib.
	AddQuotedAttrib (see page 2233)	This is function MidItems.AddQuotedAttrib.
	AddStringAttrib (see page 2233)	This is function MidItems.AddStringAttrib.
	CanAddClassHelper (see page 2234)	This is function MidItems.CanAddClassHelper.
	CreateDefaultButtonClasses (see page 2234)	This is function MidItems.CreateDefaultButtonClasses.
	DeclareSubmitForm (see page 2234)	This is function MidItems.DeclareSubmitForm.
	FindDispatcher (see page 2235)	This is function MidItems.FindDispatcher.
	FindProducer (see page 2235)	This is function MidItems.FindProducer.
	FormatColumnData (see page 2235)	This is function MidItems.FormatColumnData.
	FormatColumnHeading (see page 2236)	This is function MidItems.FormatColumnHeading.
	GetItemValuesFromDataSet (see page 2236)	This is function MidItems.GetItemValuesFromDataSet.

37.3.1 MidItems.AddBoolAttrib Function

Pascal

```
procedure AddBoolAttrib(var Attribs: string; const Attrib: string; Value: Boolean);
```

C++

```
AddBoolAttrib(UnicodeString Attribs, const UnicodeString Attrib, Boolean Value);
```

File

MidItems

Description

This is function MidItems.AddBoolAttrib.

37.3.2 MidItems.AddCustomAttrib Function

Pascal

```
procedure AddCustomAttrib(var Attribs: string; Value: string);
```

C++

```
AddCustomAttrib(UnicodeString Attribs, UnicodeString Value);
```

File

MidItems

Description

This is function MidItems.AddCustomAttrib.

37.3.3 MidItems.AddIntAttrib Function

Pascal

```
procedure AddIntAttrib(var Attribs: string; const Attrib: string; Value: Integer);
```

C++

```
AddIntAttrib(UnicodeString Attribs, const UnicodeString Attrib, int Value);
```

File

MidItems

Description

This is function MidItems.AddIntAttrib.

37.3.4 MidItems.AddQuotedAttrib Function

Pascal

```
procedure AddQuotedAttrib(var Attribs: string; const Attrib: string; const Value: string);
```

C++

```
AddQuotedAttrib(UnicodeString Attribs, const UnicodeString Attrib, const UnicodeString Value);
```

File

MidItems

Description

This is function MidItems.AddQuotedAttrib.

37.3.5 MidItems.AddStringAttrib Function

Pascal

```
procedure AddStringAttrib(var Attribs: string; const Attrib: string; const Value: string);
```

C++

```
AddStringAttrib(UnicodeString Attribs, const UnicodeString Attrib, const UnicodeString Value);
```

File

MidItems

Description

This is function MidItems.AddStringAttrib.

37.3.6 MidItems.CanAddClassHelper Function

Pascal

```
function CanAddClassHelper(AEditor: TComponent; AParent: TComponent; AClass: TClass): Boolean;
```

C++

```
Boolean CanAddClassHelper(TComponent * AEditor, TComponent * AParent, TClass AClass);
```

File

MidItems

Description

This is function MidItems.CanAddClassHelper.

37.3.7 MidItems.CreateDefaultButtonClasses Function

Pascal

```
procedure CreateDefaultButtonClasses(const Classes: array of TWebButtonClass; Container:
TWebComponentList);
```

C++

```
CreateDefaultButtonClasses(const array of TWebButtonClass Classes, TWebComponentList
Container);
```

File

MidItems

Description

This is function MidItems.CreateDefaultButtonClasses.

37.3.8 MidItems.DeclareSubmitForm Function

Pascal

```
procedure DeclareSubmitForm(Component: TComponent; XMLBroker: TXMLBroker; AddIntf:
IAddScriptElements);
```

C++

```
DeclareSubmitForm(TComponent * Component, TXMLBroker XMLBroker, IAddScriptElements AddIntf);
```

File

MidItems

Description

This is function MidItems.DeclareSubmitForm.

37.3.9 MidItems.FindDispatcher Function

Pascal

```
function FindDispatcher(Component: TComponent): IWebDispatcherAccess;
```

C++

```
IWebDispatcherAccess FindDispatcher(TComponent * Component);
```

File

MidItems

Description

This is function MidItems.FindDispatcher.

37.3.10 MidItems.FindProducer Function

Pascal

```
function FindProducer(Component: TComponent): TCustomContentProducer;
```

C++

```
TCustomContentProducer FindProducer(TComponent * Component);
```

File

MidItems

Description

This is function MidItems.FindProducer.

37.3.11 MidItems.FormatColumnData Function

Pascal

```
function FormatColumnData(Field: TWebDataDisplay; Content: string): string;
```

C++

```
UnicodeString FormatColumnData(TWebDataDisplay Field, UnicodeString Content);
```

File

MidItems

Description

This is function MidItems.FormatColumnData.

37.3.12 MidItems.FormatColumnHeading Function

Pascal

```
function FormatColumnHeading(Field: TWebDataDisplay): string;
```

C++

```
UnicodeString FormatColumnHeading(TWebDataDisplay Field);
```

File

MidItems

Description

This is function MidItems.FormatColumnHeading.

37.3.13 MidItems.GetItemValuesFromDataSet Function

Pascal

```
function GetItemValuesFromDataSet(DataSet: TDataSet; const ItemsField: string; const
ValuesField: string; var ItemsStrings: TStrings; var ValuesStrings: TStrings): Boolean;
```

C++

```
Boolean GetItemValuesFromDataSet(TDataSet DataSet, const UnicodeString ItemsField, const
UnicodeString ValuesField, TStrings ItemsStrings, TStrings ValuesStrings);
```

File

MidItems

Description

This is function MidItems.GetItemValuesFromDataSet.

37.4 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TCaptionPosition (see page 2237)	TCaptionPosition indicates the relative position of a label to the HTML control it labels.
	TFormMethod (see page 2238)	TFormMethod represents the method attribute on the <form> tag of an HTML form.
	TLayoutState (see page 2238)	This is record MidItems.TLayoutState.
	TTextAreaWrap (see page 2239)	TTextAreaWrap indicates how an HTML display control should treat items that are too wide for the physical width of the control.

37.4.1 MidItems.TCaptionPosition Enumeration

TCaptionPosition indicates the relative position of a label to the HTML control it labels.

Pascal

```
TCaptionPosition = (
  capLeft,
  capRight,
  capAbove,
  capBelow
);
```

C++

```
enum TCaptionPosition {
  capLeft,
  capRight,
  capAbove,
  capBelow
};
```

File

MidItems

Description

TCaptionPosition can have one of the following values.

Value	Meaning
capLeft	The caption appears to the left of the HTML control.
capRight	The caption appears to the right of the HTML control.
capAbove	The caption appears above the HTML control.
capBelow	The caption appears below the HTML control.

37.4.2 MidItems.TFormMethod Enumeration

TFormMethod represents the method attribute on the <form> tag of an HTML form.

Pascal

```
TFormMethod = (
  fmPost,
  fmGet
);
```

C++

```
enum TFormMethod {
  fmPost,
  fmGet
};
```

File

MidItems

Description

HTML forms include a <form> tag that distinguishes them from other types of HTML pages. This<form> tag includes a method attribute, which determines the method header of the HTTP request message that the form sends to its server. The method type corresponds to the MethodType property of the HTTP request message.

Components that generate HTML forms use TFormMethod to specify the method attribute of the <form> tag they generate, and hence the method header of the messages that they send to the server.

The following table describes the possible values:

Value	Corresponding MethodType	Meaning
fmPost	mtPost	The value of the method attribute is 'post'. The form submits messages that post displayed values to the server application. Named value pairs are passed as content fields of the HTTP message.
fmGet	mtGet	The value of the method attribute is 'get'. The form submits messages that request values it can display. Named value pairs are supplied as query fields in the URL.

See Also

TWebRequest

37.4.3 MidItems.TLayoutState Enumeration

Pascal

```
TLayoutState = (
  lsUnknown,
  lsFields,
  lsButtons
);
```

C++

```
enum TLayoutState {
```

```
    lsUnknown,
    lsFields,
    lsButtons
  };
```

File

MidItems

Description

This is record MidItems.TLayoutState.

37.4.4 MidItems.TTextAreaWrap Enumeration

TTextAreaWrap indicates how an HTML display control should treat items that are too wide for the physical width of the control.

Pascal

```
  TTextAreaWrap = (
    wrOff,
    wrPhysical,
    wrVirtual
  );
```

C++

```
  enum TTextAreaWrap {
    wrOff,
    wrPhysical,
    wrVirtual
  };
```

File

MidItems

Description

TTextAreaWrap can have one of the following values.

Value	Meaning
wrOff	The control does not wrap long lines. Only newline characters that are part of the text string cause the lines in the control to wrap.
wrPhysical	The control formats its text value by wrapping long lines so that no text is hidden. The control inserts carriage returns into the text string.
wrVirtual	The control formats its text value by wrapping long lines so that no text is hidden but does not change the actual value of the text string.

37.5 Types

The following table lists types in this documentation.

Types

Name	Description
IMidasWebDataSet (see page 2240)	This is type MidItems.IMidasWebDataSet.
TWebButtonClass (see page 2240)	This is type MidItems.TWebButtonClass.
TWebDataDisplayClass (see page 2241)	This is type MidItems.TWebDataDisplayClass.
TWebDataInputClass (see page 2241)	This is type MidItems.TWebDataInputClass.
TWebStatusClass (see page 2241)	This is type MidItems.TWebStatusClass.

37.5.1 MidItems.IMidasWebDataSet Type

Pascal

```
IMidasWebDataSet = IInetXWebDataSet;
```

C++

```
IInetXWebDataSet IMidasWebDataSet;
```

File

MidItems

Description

This is type MidItems.IMidasWebDataSet.

37.5.2 MidItems.TWebButtonClass Type

Pascal

```
TWebButtonClass = class of TWebButton;
```

C++

```
TWebButton TWebButtonClass;
```

File

MidItems

Description

This is type MidItems.TWebButtonClass.

37.5.3 MidItems.TWebDataDisplayClass Type

Pascal

```
TWebDataDisplayClass = class of TWebDataDisplay;
```

C++

```
TWebDataDisplay TWebDataDisplayClass;
```

File

MidItems

Description

This is type MidItems.TWebDataDisplayClass.

37.5.4 MidItems.TWebDataInputClass Type

Pascal

```
TWebDataInputClass = class of TWebDataInput;
```

C++

```
TWebDataInput TWebDataInputClass;
```

File

MidItems

Description

This is type MidItems.TWebDataInputClass.

37.5.5 MidItems.TWebStatusClass Type

Pascal

```
TWebStatusClass = class of TWebStatus;
```

C++

```
TWebStatus TWebStatusClass;
```

File

MidItems

Description

This is type MidItems.TWebStatusClass.

37.6 Constants

The following table lists constants in this documentation.

Constants

Name	Description
FormMethodNames (see page 2242)	This is constant MidItems.FormMethodNames.
HTMLAlignValues (see page 2243)	This is constant MidItems.HTMLAlignValues.
HTMLVAlignValues (see page 2243)	This is constant MidItems.HTMLVAlignValues.
sActionAttr (see page 2243)	This is constant MidItems.sActionAttr.
sAlignAttr (see page 2244)	This is constant MidItems.sAlignAttr.
sBgColorAttr (see page 2244)	This is constant MidItems.sBgColorAttr.
sBorderAttr (see page 2244)	This is constant MidItems.sBorderAttr.
sCellPaddingAttr (see page 2245)	This is constant MidItems.sCellPaddingAttr.
sCellSpacingAttr (see page 2245)	This is constant MidItems.sCellSpacingAttr.
sCheckedAttr (see page 2245)	This is constant MidItems.sCheckedAttr.
sClassAttr (see page 2246)	This is constant MidItems.sClassAttr.
sColsAttr (see page 2246)	This is constant MidItems.sColsAttr.
sColSpanAttr (see page 2246)	This is constant MidItems.sColSpanAttr.
sEncTypeAttr (see page 2247)	This is constant MidItems.sEncTypeAttr.
sImageHeightAttr (see page 2247)	This is constant MidItems.sImageHeightAttr.
sImageWidthAttr (see page 2247)	This is constant MidItems.sImageWidthAttr.
sMaxLengthAttr (see page 2248)	This is constant MidItems.sMaxLengthAttr.
sMethodAttr (see page 2248)	This is constant MidItems.sMethodAttr.
sMultipleAttr (see page 2248)	This is constant MidItems.sMultipleAttr.
sNameAttr (see page 2249)	This is constant MidItems.sNameAttr.
sOnClickAttr (see page 2249)	This is constant MidItems.sOnClickAttr.
sReadOnlyAttr (see page 2249)	This is constant MidItems.sReadOnlyAttr.
sRowsAttr (see page 2250)	This is constant MidItems.sRowsAttr.
sSelectedAttr (see page 2250)	This is constant MidItems.sSelectedAttr.
sSizeAttr (see page 2250)	This is constant MidItems.sSizeAttr.
sStyleAttr (see page 2251)	This is constant MidItems.sStyleAttr.
sTabIndexAttr (see page 2251)	This is constant MidItems.sTabIndexAttr.
sTypeAttr (see page 2251)	This is constant MidItems.sTypeAttr.
sVAlignAttr (see page 2252)	This is constant MidItems.sVAlignAttr.
sValueAttr (see page 2252)	This is constant MidItems.sValueAttr.
sWidthAttr (see page 2252)	This is constant MidItems.sWidthAttr.
sWrapAttr (see page 2253)	This is constant MidItems.sWrapAttr.
TextAreaWrap (see page 2253)	This is constant MidItems.TextAreaWrap.

37.6.1 MidItems.FormMethodNames Constant

Pascal

```
FormMethodNames: array[TFormMethod] of string = ('POST', 'GET');
```

C++

```
array[TFormMethod] of string FormMethodNames = ('POST', 'GET');
```

File

MidItems

Description

This is constant MidItems.FormMethodNames.

37.6.2 MidItems.HTMLAlignValues Constant

Pascal

```
HTMLAlignValues: array[THTMLAlign] of string = ('', 'left', 'right', 'center');
```

C++

```
array[THTMLAlign] of string HTMLAlignValues = ('', 'left', 'right', 'center');
```

File

MidItems

Description

This is constant MidItems.HTMLAlignValues.

37.6.3 MidItems.HTMLVAlignValues Constant

Pascal

```
HTMLVAlignValues: array[THTMLVAlign] of string = ('', 'top', 'middle', 'bottom', 'baseline');
```

C++

```
array[THTMLVAlign] of string HTMLVAlignValues = ('', 'top', 'middle', 'bottom', 'baseline');
```

File

MidItems

Description

This is constant MidItems.HTMLVAlignValues.

37.6.4 MidItems.sActionAttr Constant

Pascal

```
sActionAttr = 'action';
```

C++

```
sActionAttr = 'action';
```

File

MidItems

Description

This is constant MidItems.sActionAttr.

37.6.5 MidItems.sAlignAttr Constant

Pascal

```
sAlignAttr = 'align';
```

C++

```
sAlignAttr = 'align';
```

File

MidItems

Description

This is constant MidItems.sAlignAttr.

37.6.6 MidItems.sBgColorAttr Constant

Pascal

```
sBgColorAttr = 'bgcolor';
```

C++

```
sBgColorAttr = 'bgcolor';
```

File

MidItems

Description

This is constant MidItems.sBgColorAttr.

37.6.7 MidItems.sBorderAttr Constant

Pascal

```
sBorderAttr = 'border';
```

C++

```
sBorderAttr = 'border';
```

File

MidItems

Description

This is constant MidItems.sBorderAttr.

37.6.8 MidItems.sCellPaddingAttr Constant

Pascal

```
sCellPaddingAttr = 'cellpadding';
```

C++

```
sCellPaddingAttr = 'cellpadding';
```

File

MidItems

Description

This is constant MidItems.sCellPaddingAttr.

37.6.9 MidItems.sCellSpacingAttr Constant

Pascal

```
sCellSpacingAttr = 'cellspacing';
```

C++

```
sCellSpacingAttr = 'cellspacing';
```

File

MidItems

Description

This is constant MidItems.sCellSpacingAttr.

37.6.10 MidItems.sCheckedAttr Constant

Pascal

```
sCheckedAttr = 'checked';
```

C++

```
sCheckedAttr = 'checked';
```

File

MidItems

Description

This is constant MidItems.sCheckedAttr.

37.6.11 MidItems.sClassAttr Constant

Pascal

```
sClassAttr = 'class';
```

C++

```
sClassAttr = 'class';
```

File

MidItems

Description

This is constant MidItems.sClassAttr.

37.6.12 MidItems.sColsAttr Constant

Pascal

```
sColsAttr = 'cols';
```

C++

```
sColsAttr = 'cols';
```

File

MidItems

Description

This is constant MidItems.sColsAttr.

37.6.13 MidItems.sColSpanAttr Constant

Pascal

```
sColSpanAttr = 'colspan';
```

C++

```
sColSpanAttr = 'colspan';
```

File

MidItems

Description

This is constant MidItems.sColSpanAttr.

37.6.14 MidItems.sEncTypeAttr Constant

Pascal

```
sEncTypeAttr = 'enctype';
```

C++

```
sEncTypeAttr = 'enctype';
```

File

MidItems

Description

This is constant MidItems.sEncTypeAttr.

37.6.15 MidItems.sImageHeightAttr Constant

Pascal

```
sImageHeightAttr = 'imageheight';
```

C++

```
sImageHeightAttr = 'imageheight';
```

File

MidItems

Description

This is constant MidItems.sImageHeightAttr.

37.6.16 MidItems.sImageWidthAttr Constant

Pascal

```
sImageWidthAttr = 'imagewidth';
```

C++

```
sImageWidthAttr = 'imagewidth';
```

File

MidItems

Description

This is constant MidItems.sImageWidthAttr.

37.6.17 MidItems.sMaxLengthAttr Constant

Pascal

```
sMaxLengthAttr = 'maxlength';
```

C++

```
sMaxLengthAttr = 'maxlength';
```

File

MidItems

Description

This is constant MidItems.sMaxLengthAttr.

37.6.18 MidItems.sMethodAttr Constant

Pascal

```
sMethodAttr = 'method';
```

C++

```
sMethodAttr = 'method';
```

File

MidItems

Description

This is constant MidItems.sMethodAttr.

37.6.19 MidItems.sMultipleAttr Constant

Pascal

```
sMultipleAttr = 'multiple';
```

C++

```
sMultipleAttr = 'multiple';
```

File

MidItems

Description

This is constant MidItems.sMultipleAttr.

37.6.20 MidItems.sNameAttr Constant

Pascal

```
sNameAttr = 'name';
```

C++

```
sNameAttr = 'name';
```

File

MidItems

Description

This is constant MidItems.sNameAttr.

37.6.21 MidItems.sOnClickAttr Constant

Pascal

```
sOnClickAttr = 'onclick';
```

C++

```
sOnClickAttr = 'onclick';
```

File

MidItems

Description

This is constant MidItems.sOnClickAttr.

37.6.22 MidItems.sReadOnlyAttr Constant

Pascal

```
sReadOnlyAttr = 'readonly';
```

C++

```
sReadOnlyAttr = 'readonly';
```

File

MidItems

Description

This is constant MidItems.sReadOnlyAttr.

37.6.23 MidItems.sRowsAttr Constant

Pascal

```
sRowsAttr = 'rows';
```

C++

```
sRowsAttr = 'rows';
```

File

MidItems

Description

This is constant MidItems.sRowsAttr.

37.6.24 MidItems.sSelectedAttr Constant

Pascal

```
sSelectedAttr = 'selected';
```

C++

```
sSelectedAttr = 'selected';
```

File

MidItems

Description

This is constant MidItems.sSelectedAttr.

37.6.25 MidItems.sSizeAttr Constant

Pascal

```
sSizeAttr = 'size';
```

C++

```
sSizeAttr = 'size';
```

File

MidItems

Description

This is constant MidItems.sSizeAttr.

37.6.26 MidItems.sStyleAttr Constant

Pascal

```
sStyleAttr = 'style';
```

C++

```
sStyleAttr = 'style';
```

File

MidItems

Description

This is constant MidItems.sStyleAttr.

37.6.27 MidItems.sTabIndexAttr Constant

Pascal

```
sTabIndexAttr = 'tabindex';
```

C++

```
sTabIndexAttr = 'tabindex';
```

File

MidItems

Description

This is constant MidItems.sTabIndexAttr.

37.6.28 MidItems.sTypeAttr Constant

Pascal

```
sTypeAttr = 'type';
```

C++

```
sTypeAttr = 'type';
```

File

MidItems

Description

This is constant MidItems.sTypeAttr.

37.6.29 MidItems.sVAlignAttr Constant

Pascal

```
sVAlignAttr = 'valign';
```

C++

```
sVAlignAttr = 'valign';
```

File

MidItems

Description

This is constant MidItems.sVAlignAttr.

37.6.30 MidItems.sValueAttr Constant

Pascal

```
sValueAttr = 'value';
```

C++

```
sValueAttr = 'value';
```

File

MidItems

Description

This is constant MidItems.sValueAttr.

37.6.31 MidItems.sWidthAttr Constant

Pascal

```
sWidthAttr = 'width';
```

C++

```
sWidthAttr = 'width';
```

File

MidItems

Description

This is constant MidItems.sWidthAttr.

37.6.32 MidItems.sWrapAttr Constant

Pascal

```
sWrapAttr = 'wrap';
```

C++

```
sWrapAttr = 'wrap';
```

File

MidItems

Description

This is constant MidItems.sWrapAttr.

37.6.33 MidItems.TextAreaWrap Constant

Pascal

```
TextAreaWrap: array[TTextAreaWrap] of string = ('OFF', 'PHYSICAL', 'VIRTUAL');
```

C++

```
array[TTextAreaWrap] of string TextAreaWrap = ('OFF', 'PHYSICAL', 'VIRTUAL');
```

File

MidItems

Description

This is constant MidItems.TextAreaWrap.

38 MidProd Namespace

This is namespace MidProd.

Topics

Name	Description
Classes (see page 2257)	The following table lists classes in this documentation.
Functions (see page 2296)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 2297)	The following table lists structs, records, enums in this documentation.
Types (see page 2298)	The following table lists types in this documentation.
Constants (see page 2301)	The following table lists constants in this documentation.

Classes

	Name	Description
	TBaseInetXPageProducer (see page 2257)	TBaseInetXPageProducer is an base class for InternetExpress page producer classes.
	TCustomInetXPageProducer (see page 2275)	TCustomInetXPageProducer is the base class for page producers that display live database information in an InternetExpress application.
	TInetXPageElements (see page 2283)	This is class MidProd.TInetXPageElements.
	TInetXPageProducer (see page 2285)	TInetXPageProducer assembles an HTML document that represents live database information from an application server.
	TMidasPageProducer (see page 2295)	TMidasPageProducer assembles an HTML document that represents live database information from an application server.

Constants

Name	Description
sXMLBroker (see page 2301)	This is constant MidProd.sXMLBroker.
sXMLReadyVar (see page 2301)	This is constant MidProd.sXMLReadyVar.

Functions

	Name	Description
	AddStdIncludes (see page 2296)	This is function MidProd.AddStdIncludes.
	PathInfoToRelativePath (see page 2296)	This is function MidProd.PathInfoToRelativePath.

Structs, Records, Enums

	Name	Description
	TStdInclude (see page 2297)	This is record MidProd.TStdInclude.

Types

Name	Description
TBaseMidasPageProducer (see page 2298)	This is type MidProd.TBaseMidasPageProducer.
TCustomMidasPageProducer (see page 2298)	This is type MidProd.TCustomMidasPageProducer.
TMidasPageElements (see page 2299)	This is type MidProd.TMidasPageElements.
TStdIncludes (see page 2299)	This is type MidProd.TStdIncludes.
TXMLDataEvent (see page 2299)	TXMLDataEvent is the type of event handlers that occur when an InternetExpress page producer adds an XML data packet to its HTML document.

38.1 Classes

The following table lists classes in this documentation.

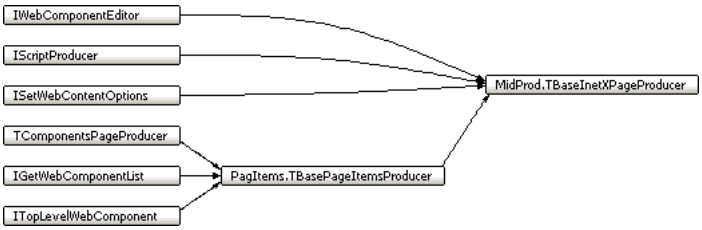
Classes

	Name	Description
	TBaseInetXPageProducer (see page 2257)	TBaseInetXPageProducer is an base class for InternetExpress page producer classes.
	TCustomInetXPageProducer (see page 2275)	TCustomInetXPageProducer is the base class for page producers that display live database information in an InternetExpress application.
	TInetXPageElements (see page 2283)	This is class MidProd.TInetXPageElements.
	TInetXPageProducer (see page 2285)	TInetXPageProducer assembles an HTML document that represents live database information from an application server.
	TMidasPageProducer (see page 2295)	TMidasPageProducer assembles an HTML document that represents live database information from an application server.

38.1.1 TBaseInetXPageProducer Class

TBaseInetXPageProducer is an base class for InternetExpress page producer classes.

Class Hierarchy



Pascal

```
TBaseInetXPageProducer = class(TBasePageItemsProducer, IWebComponentEditor, IScriptProducer,
ISetWebContentOptions);
```

C++

```
class TBaseInetXPageProducer : public TBasePageItemsProducer, public IWebComponentEditor,
public IScriptProducer, public ISetWebContentOptions;
```

File

MidProd

Description

Do not create instances of TBaseInetXPageProducer. Instead, use one of its descendant classes, such as TInetXPageProducer (see page 2285).

TBaseInetXPageProducer introduces the logic for working with an XML broker to obtain database information and for translating the special HTML-transparent tags that are used in InternetExpress applications. Unlike its immediate descendant (TCustomInetXPageProducer (see page 2275)) however, TBaseInetXPageProducer does not initialize the HTMLDoc property to the default template for an InternetExpress application, although it dynamically generates the InternetExpress-specific tags.

You can use TBaseInetXPageProducer as a base class when writing a page producer that does not use the default template, but

which translates InternetExpress tags into HTML and javascript.

See Also

TInetXPageProducer (see page 2285)

TXMLBroker

38.1.1.1 TBaseInetXPageProducer Members

The following tables list the members exposed by TBaseInetXPageProducer.

Protected Methods

	Name	Description
	AddIncludes (see page 2261)	This is AddIncludes, a member of class TBaseInetXPageProducer.
	AddScriptComponents (see page 2261)	This is AddScriptComponents, a member of class TBaseInetXPageProducer.
	CanAddClass (see page 2261)	Indicates whether a specified class can be used as a child Web item.
	CreatePageElements (see page 2262)	This is CreatePageElements, a member of class TBaseInetXPageProducer.
	CreateScriptManager (see page 2262)	This is CreateScriptManager, a member of class TBaseInetXPageProducer.
	DeclareFunctions (see page 2263)	This is DeclareFunctions, a member of class TBaseInetXPageProducer.
	DeclareIncludes (see page 2263)	This is DeclareIncludes, a member of class TBaseInetXPageProducer.
	DeclareRowSets (see page 2263)	This is DeclareRowSets, a member of class TBaseInetXPageProducer.
	DeclareScript (see page 2263)	This is DeclareScript, a member of class TBaseInetXPageProducer.
	DeclareStyles (see page 2263)	This is DeclareStyles, a member of class TBaseInetXPageProducer.
	DeclareWarnings (see page 2264)	This is DeclareWarnings, a member of class TBaseInetXPageProducer.
	DeclareXMLDocuments (see page 2264)	This is DeclareXMLDocuments, a member of class TBaseInetXPageProducer.
	DeclareXMLIslands (see page 2264)	This is DeclareXMLIslands, a member of class TBaseInetXPageProducer.
	DoAfterGetContent (see page 2264)	This is DoAfterGetContent, a member of class TBaseInetXPageProducer.
	DoAfterGetXMLData (see page 2265)	This is DoAfterGetXMLData, a member of class TBaseInetXPageProducer.
	DoBeforeGetContent (see page 2265)	This is DoBeforeGetContent, a member of class TBaseInetXPageProducer.
	DoBeforeGetXMLData (see page 2265)	This is DoBeforeGetXMLData, a member of class TBaseInetXPageProducer.
	DoTagEvent (see page 2265)	This is DoTagEvent, a member of class TBaseInetXPageProducer.
	GeneratePageElements (see page 2266)	This is GeneratePageElements, a member of class TBaseInetXPageProducer.
	GetChildren (see page 2968)	Enumerates all child components.
	GetComponentList (see page 2968)	This is GetComponentList, a member of class TBasePageItemsProducer.
	GetContentOptions (see page 2267)	This is GetContentOptions, a member of class TBaseInetXPageProducer.
	GetDefaultComponentList (see page 2969)	This is GetDefaultComponentList, a member of class TBasePageItemsProducer.
	GetDefaultTemplate (see page 2267)	This is GetDefaultTemplate, a member of class TBaseInetXPageProducer.
	GetScriptManager (see page 2268)	This is GetScriptManager, a member of class TBaseInetXPageProducer.
	GetUseXMLIslands (see page 2268)	This is GetUseXMLIslands, a member of class TBaseInetXPageProducer.
	GetXMLDelta (see page 2268)	This is GetXMLDelta, a member of class TBaseInetXPageProducer.
	GetXMLErrors (see page 2269)	This is GetXMLErrors, a member of class TBaseInetXPageProducer.
	ImplCanAddClass (see page 2269)	This is ImplCanAddClass, a member of class TBaseInetXPageProducer.
	SetChildOrder (see page 2969)	Sets the visual ordering of a child control.
	SetWebContentOptions (see page 2270)	This is SetWebContentOptions, a member of class TBaseInetXPageProducer.

Protected Properties

	Name	Description
	PageElements (see page 2271)	This is PageElements, a member of class TBaseInetXPageProducer.
	ScriptManager (see page 2272)	This is ScriptManager, a member of class TBaseInetXPageProducer.

Public Events

	Name	Description
	OnAfterGetContent (see page 2273)	Occurs after the page producer generates its HTML document.

	OnAfterGetXMLData (see page 2274)	Occurs after the page producer translates fetches an XML data packet from an XML broker.
	OnBeforeGetContent (see page 2274)	Occurs when the Content method is called.
	OnBeforeGetXMLData (see page 2275)	Occurs before the page producer fetches an XML data packet from an XML broker.

Public Methods

	Name	Description
V	ContentFromStream (see page 2262)	Returns the content produced in response to input read from a stream.
V	Create (see page 2259)	Creates an instance of a TBaseInetXPageProducer (see page 2257).
V	Destroy (see page 2259)	Destroys the instance of the TBasePageItemsProducer.
V	FindXMLBroker (see page 2265)	Returns the XML broker that supplies the data packet specified by a DATAPACKET tag.
V	FindXMLBrokerName (see page 2266)	Returns the name of an XML broker given the parameters in a DATAPACKET tag..
V	GetRequiredTags (see page 2267)	Returns the default set of HTML-transparent tags for an InternetExpress page producer.
V	GetXMLData (see page 2268)	Obtains an XML data packet from a specified XML broker.
	SetIncludePathURL (see page 2269)	Assigns a value to the IncludePathURL (see page 2271) property.
	SetStyles (see page 2269)	Assigns a value to the Styles (see page 2272) property.
	SetStylesFile (see page 2270)	Assigns a value to the StylesFile (see page 2273) property.

Public Properties

	Name	Description
	EnableXMLIslands (see page 2271)	Specifies whether XML data packets should be declared using XML islands if the requesting browser supports them.
	IncludePathURL (see page 2271)	Indicates the URL where the javascript libraries are installed.
	Styles (see page 2272)	Defines the styles that can be used by the page producer and its Web items.
	StylesFile (see page 2273)	Specifies a file containing the individual styles for the page producer's style sheet.
R	WebPageItems (see page 2969)	Maintains a list of components that generate HTML content for the producer.

38.1.1.2 TBaseInetXPageProducer.Create Constructor

Creates an instance of a TBaseInetXPageProducer (see page 2257).

Pascal

```
constructor Create(AComponent: TComponent); override;
```

C++

```
virtual __fastcall TBaseInetXPageProducer(TComponent * AComponent);
```

Description

Call Create to instantiate a TBaseInetXPageProducer (see page 2257) component at runtime if it was not created at design time.

AOwner is the component, typically the Web module, that becomes the value of this page producer's Owner property. The Owner is responsible for freeing the TBaseInetXPageProducer (see page 2257) instance.

See Also

Owner

38.1.1.3 TBaseInetXPageProducer.Destroy Destructor

Destroys the instance of the TBasePageItemsProducer.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TBaseInetXPageProducer();
```

Description

Do not call Destroy directly. Instead call Free, which checks that the TBasePageItemsProducer reference is not nil before calling Destroy.

See Also

Create (see page 2967)

System::TObject::Free

38.1.1.4 TBaseInetXPageProducer Methods

The methods of the TBaseInetXPageProducer class are listed here.

Protected Methods

	Name	Description
	AddIncludes (see page 2261)	This is AddIncludes, a member of class TBaseInetXPageProducer.
	AddScriptComponents (see page 2261)	This is AddScriptComponents, a member of class TBaseInetXPageProducer.
	CanAddClass (see page 2261)	Indicates whether a specified class can be used as a child Web item.
	CreatePageElements (see page 2262)	This is CreatePageElements, a member of class TBaseInetXPageProducer.
	CreateScriptManager (see page 2262)	This is CreateScriptManager, a member of class TBaseInetXPageProducer.
	DeclareFunctions (see page 2263)	This is DeclareFunctions, a member of class TBaseInetXPageProducer.
	DeclareIncludes (see page 2263)	This is DeclareIncludes, a member of class TBaseInetXPageProducer.
	DeclareRowSets (see page 2263)	This is DeclareRowSets, a member of class TBaseInetXPageProducer.
	DeclareScript (see page 2263)	This is DeclareScript, a member of class TBaseInetXPageProducer.
	DeclareStyles (see page 2263)	This is DeclareStyles, a member of class TBaseInetXPageProducer.
	DeclareWarnings (see page 2264)	This is DeclareWarnings, a member of class TBaseInetXPageProducer.
	DeclareXMLDocuments (see page 2264)	This is DeclareXMLDocuments, a member of class TBaseInetXPageProducer.
	DeclareXMLIslands (see page 2264)	This is DeclareXMLIslands, a member of class TBaseInetXPageProducer.
	DoAfterGetContent (see page 2264)	This is DoAfterGetContent, a member of class TBaseInetXPageProducer.
	DoAfterGetXMLData (see page 2265)	This is DoAfterGetXMLData, a member of class TBaseInetXPageProducer.
	DoBeforeGetContent (see page 2265)	This is DoBeforeGetContent, a member of class TBaseInetXPageProducer.
	DoBeforeGetXMLData (see page 2265)	This is DoBeforeGetXMLData, a member of class TBaseInetXPageProducer.
	DoTagEvent (see page 2265)	This is DoTagEvent, a member of class TBaseInetXPageProducer.
	GeneratePageElements (see page 2266)	This is GeneratePageElements, a member of class TBaseInetXPageProducer.
	GetContentOptions (see page 2267)	This is GetContentOptions, a member of class TBaseInetXPageProducer.
	GetDefaultTemplate (see page 2267)	This is GetDefaultTemplate, a member of class TBaseInetXPageProducer.
	GetScriptManager (see page 2268)	This is GetScriptManager, a member of class TBaseInetXPageProducer.
	GetUseXMLIslands (see page 2268)	This is GetUseXMLIslands, a member of class TBaseInetXPageProducer.
	GetXMLDelta (see page 2268)	This is GetXMLDelta, a member of class TBaseInetXPageProducer.
	GetXMLErrors (see page 2269)	This is GetXMLErrors, a member of class TBaseInetXPageProducer.
	ImplCanAddClass (see page 2269)	This is ImplCanAddClass, a member of class TBaseInetXPageProducer.
	SetWebContentOptions (see page 2270)	This is SetWebContentOptions, a member of class TBaseInetXPageProducer.

Public Methods

	Name	Description
	ContentFromStream (see page 2262)	Returns the content produced in response to input read from a stream.

	FindXMLBroker (see page 2265)	Returns the XML broker that supplies the data packet specified by a DATAPACKET tag.
	FindXMLBrokerName (see page 2266)	Returns the name of an XML broker given the parameters in a DATAPACKET tag..
	GetRequiredTags (see page 2267)	Returns the default set of HTML-transparent tags for an InternetExpress page producer.
	GetXMLData (see page 2268)	Obtains an XML data packet from a specified XML broker.
	SetIncludePathURL (see page 2269)	Assigns a value to the IncludePathURL (see page 2271) property.
	SetStyles (see page 2269)	Assigns a value to the Styles (see page 2272) property.
	SetStylesFile (see page 2270)	Assigns a value to the StylesFile (see page 2273) property.

38.1.1.4.1 TBaseInetXPageProducer.AddIncludes Method

Pascal

```
procedure AddIncludes; virtual;
```

C++

```
virtual __fastcall AddIncludes();
```

Description

This is AddIncludes, a member of class TBaseInetXPageProducer.

38.1.1.4.2 TBaseInetXPageProducer.AddScriptComponents Method

Pascal

```
procedure AddScriptComponents; virtual;
```

C++

```
virtual __fastcall AddScriptComponents();
```

Description

This is AddScriptComponents, a member of class TBaseInetXPageProducer.

38.1.1.4.3 TBaseInetXPageProducer.CanAddClass Method

Indicates whether a specified class can be used as a child Web item.

Pascal

```
function CanAddClass(AParent: TComponent; AClass: TClass): Boolean;
```

C++

```
__fastcall Boolean CanAddClass(TComponent * AParent, TClass AClass);
```

Description

The Web page editor calls CanAddClass to determine what Web items can be added to supply content for the Web item that implements this interface.

AParent is the Web item that will act as a parent to the added Web item.

AClass is the class of the proposed child Web item.

CanAddClass returns true if the Web item can use content generated by a component of type AClass, false otherwise.

38.1.1.4.4 TBaseInetXPageProducer.ContentFromStream Method

Returns the content produced in response to input read from a stream.

Pascal

```
function ContentFromStream(InStream: TStream): string; override;
```

C++

```
virtual __fastcall UnicodeString ContentFromStream(TStream InStream);
```

Description

Call ContentFromStream to retrieve the content produced by the content producer based on the information provided by the Stream parameter. ContentFromStream

1. Generates an OnBeforeGetContent (see page 2274) event.
2. Translates the HTML template that is the value of Stream, using information produced by the items listed in the WebPageItems property.
3. Generates an OnAfterGetContent (see page 2273) event.
4. Returns the HTML produced in step 2.

See Also

Content

OnBeforeGetContent (see page 2274)

WebPageItems (see page 2969)

OnAfterGetContent (see page 2273)

38.1.1.4.5 TBaseInetXPageProducer.CreatePageElements Method

Pascal

```
function CreatePageElements: TInetxPageElements; virtual;
```

C++

```
virtual __fastcall TInetxPageElements CreatePageElements();
```

Description

This is CreatePageElements, a member of class TBaseInetXPageProducer.

38.1.1.4.6 TBaseInetXPageProducer.CreateScriptManager Method

Pascal

```
function CreateScriptManager(Options: TWebContentOptions): TObject; virtual;
```

C++

```
virtual __fastcall TObject * CreateScriptManager(TWebContentOptions Options);
```

Description

This is CreateScriptManager, a member of class TBaseInetXPageProducer.

38.1.1.4.7 TBaseInetXPageProducer.DeclareFunctions Method

Pascal

```
function DeclareFunctions: string; virtual;
```

C++

```
virtual __fastcall UnicodeString DeclareFunctions();
```

Description

This is DeclareFunctions, a member of class TBaseInetXPageProducer.

38.1.1.4.8 TBaseInetXPageProducer.DeclareIncludes Method

Pascal

```
function DeclareIncludes: string; virtual;
```

C++

```
virtual __fastcall UnicodeString DeclareIncludes();
```

Description

This is DeclareIncludes, a member of class TBaseInetXPageProducer.

38.1.1.4.9 TBaseInetXPageProducer.DeclareRowSets Method

Pascal

```
function DeclareRowSets: string; virtual;
```

C++

```
virtual __fastcall UnicodeString DeclareRowSets();
```

Description

This is DeclareRowSets, a member of class TBaseInetXPageProducer.

38.1.1.4.10 TBaseInetXPageProducer.DeclareScript Method

Pascal

```
function DeclareScript(XMLIslands: Boolean): string; virtual;
```

C++

```
virtual __fastcall UnicodeString DeclareScript(Boolean XMLIslands);
```

Description

This is DeclareScript, a member of class TBaseInetXPageProducer.

38.1.1.4.11 TBaseInetXPageProducer.DeclareStyles Method

Pascal

```
function DeclareStyles: string; virtual;
```

C++

```
virtual __fastcall UnicodeString DeclareStyles();
```

Description

This is DeclareStyles, a member of class TBaseInetXPageProducer.

38.1.1.4.12 TBaseInetXPageProducer.DeclareWarnings Method

Pascal

```
function DeclareWarnings(Warnings: TStrings): string; virtual;
```

C++

```
virtual __fastcall UnicodeString DeclareWarnings(TStrings Warnings);
```

Description

This is DeclareWarnings, a member of class TBaseInetXPageProducer.

38.1.1.4.13 TBaseInetXPageProducer.DeclareXMLDocuments Method

Pascal

```
function DeclareXMLDocuments: string; virtual;
```

C++

```
virtual __fastcall UnicodeString DeclareXMLDocuments();
```

Description

This is DeclareXMLDocuments, a member of class TBaseInetXPageProducer.

38.1.1.4.14 TBaseInetXPageProducer.DeclareXMLIslands Method

Pascal

```
function DeclareXMLIslands: string; virtual;
```

C++

```
virtual __fastcall UnicodeString DeclareXMLIslands();
```

Description

This is DeclareXMLIslands, a member of class TBaseInetXPageProducer.

38.1.1.4.15 TBaseInetXPageProducer.DoAfterGetContent Method

Pascal

```
procedure DoAfterGetContent;
```

C++

```
__fastcall DoAfterGetContent();
```

Description

This is DoAfterGetContent, a member of class TBaseInetXPageProducer.

38.1.1.4.16 TBaseInetXPageProducer.DoAfterGetXMLData Method

Pascal

```
procedure DoAfterGetXMLData(XMLBroker: TXMLBroker; var OwnerData: OleVariant);
```

C++

```
__fastcall DoAfterGetXMLData(TXMLBroker XMLBroker, OleVariant OwnerData);
```

Description

This is DoAfterGetXMLData, a member of class TBaseInetXPageProducer.

38.1.1.4.17 TBaseInetXPageProducer.DoBeforeGetContent Method

Pascal

```
procedure DoBeforeGetContent;
```

C++

```
__fastcall DoBeforeGetContent();
```

Description

This is DoBeforeGetContent, a member of class TBaseInetXPageProducer.

38.1.1.4.18 TBaseInetXPageProducer.DoBeforeGetXMLData Method

Pascal

```
procedure DoBeforeGetXMLData(XMLBroker: TXMLBroker; var OwnerData: OleVariant);
```

C++

```
__fastcall DoBeforeGetXMLData(TXMLBroker XMLBroker, OleVariant OwnerData);
```

Description

This is DoBeforeGetXMLData, a member of class TBaseInetXPageProducer.

38.1.1.4.19 TBaseInetXPageProducer.DoTagEvent Method

Pascal

```
procedure DoTagEvent(Tag: TTag; const TagString: string; TagParams: TStrings; var ReplaceText:
string); override;
```

C++

```
virtual __fastcall DoTagEvent(TTag Tag, const UnicodeString TagString, TStrings TagParams,
UnicodeString ReplaceText);
```

Description

This is DoTagEvent, a member of class TBaseInetXPageProducer.

38.1.1.4.20 TBaseInetXPageProducer.FindXMLBroker Method

Returns the XML broker that supplies the data packet specified by a DATAPACKET tag.

Pascal

```
function FindXMLBroker(TagParams: TStrings): TXMLBroker; virtual;
```

C++

```
virtual __fastcall TXMLBroker FindXMLBroker(TStrings TagParams);
```

Description

The page producer translates HTML-transparent tags of the form

<#DATAPACKET XMLBroker=BrokerName>

into the XML returned by the XML broker. To do so, it calls FindXMLBroker to locate the XML broker specified by the XMLBroker parameter.

TagParams is a set of Name=Value pairs. This comes from the parameter portion of the DATAPACKET tag.

FindXMLBroker returns the XML broker with the name specified by the XMLBroker parameter. If TagParams does not contain a string of the form XMLBroker=BrokerName, FindXMLBroker returns nil (Delphi) or NULL (C++).

See Also

FindXMLBrokerName (see page 2266)

GetXMLData (see page 2268)

38.1.1.4.21 TBaseInetXPageProducer.FindXMLBrokerName Method

Returns the name of an XML broker given the parameters in a DATAPACKET tag..

Pascal

```
function FindXMLBrokerName(TagParams: TStrings): string; virtual;
```

C++

```
virtual __fastcall UnicodeString FindXMLBrokerName(TStrings TagParams);
```

Description

The page producer translates HTML-transparent tags of the form

<#DATAPACKET XMLBroker=BrokerName>

into the XML returned by the XML broker. To do so, it calls FindXMLBrokerName to parse the parameters for the value of the XMLBroker parameter.

TagParams is a set of Name=Value pairs. This comes from the parameter portion of the DATAPACKET tag.

FindXMLBrokerName returns the value string from the XMLBroker parameter. If TagParams does not contain a string of the form XMLBroker=BrokerName, FindXMLBrokerName returns an empty string.

See Also

FindXMLBroker (see page 2265)

GetXMLData (see page 2268)

38.1.1.4.22 TBaseInetXPageProducer.GeneratePageElements Method

Pascal

```
procedure GeneratePageElements; virtual;
```

C++

```
virtual __fastcall GeneratePageElements();
```

Description

This is GeneratePageElements, a member of class TBaseInetXPageProducer.

38.1.1.4.23 TBaseInetXPageProducer.GetContentOptions Method

Pascal

```
function GetContentOptions(var Owned: Boolean): TWebContentOptions; override;
```

C++

```
virtual __fastcall TWebContentOptions GetContentOptions(Boolean Owned);
```

Description

This is GetContentOptions, a member of class TBaseInetXPageProducer.

38.1.1.4.24 TBaseInetXPageProducer.GetDefaultTemplate Method

Pascal

```
function GetDefaultTemplate: string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetDefaultTemplate();
```

Description

This is GetDefaultTemplate, a member of class TBaseInetXPageProducer.

38.1.1.4.25 TBaseInetXPageProducer.GetRequiredTags Method

Returns the default set of HTML-transparent tags for an InternetExpress page producer.

Pascal

```
class function GetRequiredTags: string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetRequiredTags();
```

Description

GetRequiredTags dynamically generates the InternetExpress-specific tags that form the basis of an InternetExpress page producer's default template.

GetRequiredTags returns the string

```
<#INCLUDES> <#STYLES> <#WARNINGS> <#FORMS> <#SCRIPT>
```

See Also

HTMLDoc (see page 2281)

Content (see page 2279)

38.1.1.4.26 TBaseInetXPageProducer.GetScriptManager Method

Pascal

```
function GetScriptManager: IScriptManager;
```

C++

```
__fastcall IScriptManager GetScriptManager();
```

Description

This is GetScriptManager, a member of class TBaseInetXPageProducer.

38.1.1.4.27 TBaseInetXPageProducer.GetUseXMLIslands Method

Pascal

```
function GetUseXMLIslands: Boolean; virtual;
```

C++

```
virtual __fastcall Boolean GetUseXMLIslands();
```

Description

This is GetUseXMLIslands, a member of class TBaseInetXPageProducer.

38.1.1.4.28 TBaseInetXPageProducer.GetXMLData Method

Obtains an XML data packet from a specified XML broker.

Pascal

```
function GetXMLData(XMLBroker: TXMLBroker; XMLOptions: TXMLOptions): string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetXMLData(TXMLBroker XMLBroker, TXMLOptions XMLOptions);
```

Description

GetXMLData first generates an OnBeforeGetXMLData (see page 2275) event. It then calls the RequestRecords method of the XML broker specified by XMLBroker, passing in the formatting options specified by XMLOptions. Finally, it generates an OnAfterGetXMLData (see page 2274) event.

GetXMLData returns the XML data packet that XMLBroker supplies.

See Also

OnAfterGetXMLData (see page 2274)

OnBeforeGetXMLData (see page 2275)

Content

RequestRecords

38.1.1.4.29 TBaseInetXPageProducer.GetXMLDelta Method

Pascal

```
function GetXMLDelta(XMLBroker: TXMLBroker): string;
```

C++

```
__fastcall UnicodeString GetXMLDelta(TXMLBroker XMLBroker);
```

Description

This is GetXMLDelta, a member of class TBaseInetXPageProducer.

38.1.1.4.30 TBaseInetXPageProducer.GetXMLErrors Method

Pascal

```
function GetXMLErrors(XMLBroker: TXMLBroker): string;
```

C++

```
__fastcall UnicodeString GetXMLErrors(TXMLBroker XMLBroker);
```

Description

This is GetXMLErrors, a member of class TBaseInetXPageProducer.

38.1.1.4.31 TBaseInetXPageProducer.ImplCanAddClass Method

Pascal

```
function ImplCanAddClass(AParent: TComponent; AClass: TClass): Boolean; virtual;
```

C++

```
virtual __fastcall Boolean ImplCanAddClass(TComponent * AParent, TClass AClass);
```

Description

This is ImplCanAddClass, a member of class TBaseInetXPageProducer.

38.1.1.4.32 TBaseInetXPageProducer.SetIncludePathURL Method

Assigns a value to the IncludePathURL (see page 2271) property.

Pascal

```
procedure SetIncludePathURL(const Value: string);
```

C++

```
__fastcall SetIncludePathURL(const UnicodeString Value);
```

Description

SetIncludePathURL is the write implementation of the IncludePathURL (see page 2271) property.

See Also

IncludePathURL (see page 2271)

38.1.1.4.33 TBaseInetXPageProducer.SetStyles Method

Assigns a value to the Styles (see page 2272) property.

Pascal

```
procedure SetStyles(Value: TStrings);
```

C++

```
__fastcall SetStyles(TStrings Value);
```

Description

SetStyles is the write implementation of the Styles (see page 2272) property.

See Also

Styles (see page 2272)

38.1.1.4.34 TBaseInetXPageProducer.SetStylesFile Method

Assigns a value to the StylesFile (see page 2273) property.

Pascal

```
procedure SetStylesFile(const Value: TFileName);
```

C++

```
__fastcall SetStylesFile(const TFileName Value);
```

Description

SetStylesFile is the write implementation of the StylesFile (see page 2273) property.

See Also

StylesFile (see page 2273)

38.1.1.4.35 TBaseInetXPageProducer.SetWebContentOptions Method

Pascal

```
procedure SetWebContentOptions(AOptions: TWebContentOptions);
```

C++

```
__fastcall SetWebContentOptions(TWebContentOptions AOptions);
```

Description

This is SetWebContentOptions, a member of class TBaseInetXPageProducer.

38.1.1.5 TBaseInetXPageProducer Properties

The properties of the TBaseInetXPageProducer class are listed here.

Protected Properties

	Name	Description
	PageElements (see page 2271)	This is PageElements, a member of class TBaseInetXPageProducer.
	ScriptManager (see page 2272)	This is ScriptManager, a member of class TBaseInetXPageProducer.

Public Properties

	Name	Description
	EnableXMLIslands (see page 2271)	Specifies whether XML data packets should be declared using XML islands if the requesting browser supports them.
	IncludePathURL (see page 2271)	Indicates the URL where the javascript libraries are installed.
	Styles (see page 2272)	Defines the styles that can be used by the page producer and its Web items.
	StylesFile (see page 2273)	Specifies a file containing the individual styles for the page producer's style sheet.

38.1.1.5.1 TBaseInetXPageProducer.EnableXMLIslands Property

Specifies whether XML data packets should be declared using XML islands if the requesting browser supports them.

Pascal

```
property EnableXMLIslands: Boolean;
```

C++

```
__property Boolean EnableXMLIslands;
```

Description

Set EnableXMLIslands to indicate whether the XML in the generated HTML document should be declared using XML islands. XML islands are only used if the document is created in response to a request message whose headers indicate that the browser supports XML islands.

Set EnableXMLIslands to false to prevent the page producer from declaring XML islands even if the requesting browser supports them.

See Also

GetXMLData (see page 2268)

Content

38.1.1.5.2 TBaseInetXPageProducer.IncludePathURL Property

Indicates the URL where the javascript libraries are installed.

Pascal

```
property IncludePathURL: string;
```

C++

```
__property UnicodeString IncludePathURL;
```

Description

Set IncludePathURL to indicate where the javascript libraries are installed. The page producer uses this value to create HTML statements that include the libraries in the generated HTML document. These statements have the form

```
<SCRIPT language=Javascript type="text/javascript" SRC="IncludePathURL/xmldb.js"> </SCRIPT>
```

The javascript libraries define classes and methods that allow the HTML document to interpret XML data packets if the Web browser is not XML-enabled, and that enable controls in the document to navigate the data, post changes, and apply updates.

See Also

SetIncludePathURL (see page 2269)

Content

Creating Web Pages with an InternetExpress Page Producer

38.1.1.5.3 TBaseInetXPageProducer.PageElements Property

Pascal

```
property PageElements: TInetxPageElements;
```

C++

```
__property TInetxPageElements PageElements;
```

Description

This is PageElements, a member of class TBaseInetXPageProducer.

38.1.1.5.4 TBaseInetXPageProducer.ScriptManager Property

Pascal

```
property ScriptManager: IScriptManager;
```

C++

```
__property IScriptManager ScriptManager;
```

Description

This is ScriptManager, a member of class TBaseInetXPageProducer.

38.1.1.5.5 TBaseInetXPageProducer.Styles Property

Defines the styles that can be used by the page producer and its Web items.

Pascal

```
property Styles: TStrings;
```

C++

```
__property TStrings Styles;
```

Description

Use Styles to define a style sheet for the HTML document produced by TBaseInetXPageProducer (see page 2257). Each string in the TStrings object specifies a specific style by giving a style selector followed by a set of attributes in curly braces. These definitions can define styles for standard HTML elements such as

```
H2 B  {color: red}
```

or they can define styles that you name, such as

```
.MyStyle  {font-family: arial; font-weight: bold; font-size: 18px }
```

Note that user-defined style names must begin with a dot.

Styles for standard elements are applied automatically to HTML that uses the tag or tags that form the style selector. User-defined styles can be applied by setting the StyleRule property of a Web item to the user-defined style name that forms the selector.

Note: If you are sharing a set of styles between multiple page producers or multiple applications, you can define the styles in a file. To define the styles in a file, use the StylesFile (see page 2273) property instead. If StylesFile (see page 2273) is defined, it takes precedence over the Styles property.

See Also

SetStyles (see page 2269)

StylesFile (see page 2273)

Content

Customizing the InternetExpress Page Producer Template

Setting Web Item Properties

38.1.1.5.6 TBaseInetXPageProducer.StylesFile Property

Specifies a file containing the individual styles for the page producer's style sheet.

Pascal

```
property StylesFile: TFileName;
```

C++

```
__property TFileName StylesFile;
```

Description

Set StylesFile to the name of a file containing the style definitions used by the page producer. The page producer uses these style definitions to declare a style sheet in the HTML document it generates.

Styles (see page 2272) for standard elements are applied automatically to HTML in the document. User-defined styles can be applied by setting the StyleRule property of a Web item to the user-defined style name that forms the selector of a style definition.

Note: If the StylesFile property is not set, you can specify the style definitions using the Styles (see page 2272) property instead.

See Also

SetStylesFile (see page 2270)

Styles (see page 2272)

Content

Customizing the InternetExpress Page Producer Template

Setting Web Item Properties

38.1.1.6 TBaseInetXPageProducer Events

The events of the TBaseInetXPageProducer class are listed here.

Public Events

	Name	Description
	OnAfterGetContent (see page 2273)	Occurs after the page producer generates its HTML document.
	OnAfterGetXMLData (see page 2274)	Occurs after the page producer translates fetches an XML data packet from an XML broker.
	OnBeforeGetContent (see page 2274)	Occurs when the Content method is called.
	OnBeforeGetXMLData (see page 2275)	Occurs before the page producer fetches an XML data packet from an XML broker.

38.1.1.6.1 TBaseInetXPageProducer.OnAfterGetContent Event

Occurs after the page producer generates its HTML document.

Pascal

```
property OnAfterGetContent: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnAfterGetContent;
```

Description

Write an OnAfterGetContent event handler to take specific action immediately after the page producer generates an HTML document. This event occurs immediately before the Content method returns the generated document. Typically it is used in applications where the page producer automatically responds to messages as the Provider (see page 3073) property of an action item or the Response producer property of an XML broker.

See Also

OnBeforeGetContent (see page 2274)

Content

Producer

38.1.1.6.2 TBaseInetXPageProducer.OnAfterGetXMLData Event

Occurs after the page producer translates fetches an XML data packet from an XML broker.

Pascal

```
property OnAfterGetXMLData: TXMLDataEvent;
```

C++

```
__property TXMLDataEvent OnAfterGetXMLData;
```

Description

Write an OnAfterGetXMLData event handler to take specific action immediately after the page producer obtains an XML data packet from an XML broker. Typically, the OnAfterGetXMLData event handler responds to any values returned by the provider on the application server as the OwnerData parameter.

When the page producer generates its HTML document (that is, when the Content method executes), it adds code to define the XML data packets manipulated in the document. To fetch these XML data packets, the page producer calls an XML broker's RequestRecords method. OnAfterGetXMLData occurs after the page producer has returned an XML data packet but before it is assigned as part of the emerging content.

See Also

OnBeforeGetXMLData (see page 2275)

GetXMLData (see page 2268)

OnHTMLTag

Content

38.1.1.6.3 TBaseInetXPageProducer.OnBeforeGetContent Event

Occurs when the Content method is called.

Pascal

```
property OnBeforeGetContent: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnBeforeGetContent;
```

Description

Write an OnBeforeGetContent event handler to take specific action immediately before the page producer generates an HTML document. This event occurs when the Content method is called before the page producer starts assembling its HTML

document. Typically it is used in applications where the page producer automatically responds to messages as the Provider (see page 3073) property of an action item or the Response producer property of an XML broker.

See Also

OnAfterGetContent (see page 2273)

Content

Producer

38.1.1.6.4 TBaseInetXPageProducer.OnBeforeGetXMLData Event

Occurs before the page producer fetches an XML data packet from an XML broker.

Pascal

```
property OnBeforeGetXMLData: TXMLDataEvent;
```

C++

```
__property TXMLDataEvent OnBeforeGetXMLData;
```

Description

Write an OnBeforeGetXMLData event handler to take specific action immediately before the page producer requests an XML data packet from an XML broker. Typically, the OnBeforeGetXMLData event handler encodes any values for the provider on the application server as the OwnerData parameter or initializes properties on the XML broker that affect the data packet.

When the page producer generates its HTML document (that is, when the Content method executes), it adds code to define the XML data packets manipulated in the document. To fetch these XML data packets, the page producer calls an XML broker's RequestRecords method. OnBeforeGetXMLData occurs immediately before the page producer requests an XML data packet. Any value assigned to the OwnerData parameter is passed to the XML broker's OnRequestRecords event handler, and from there to the provider's OnBeforeGetRecords event handler.

See Also

OnAfterGetXMLData (see page 2274)

OnRequestRecords

GetXMLData (see page 2268)

Content

OnHTMLTag

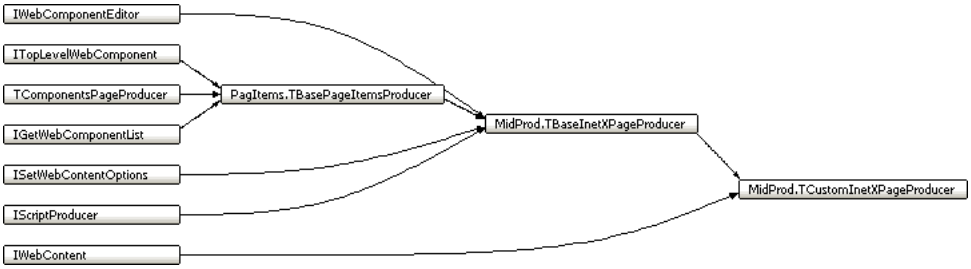
MaxRecords

Params

38.1.2 TCustomInetXPageProducer Class

TCustomInetXPageProducer is the base class for page producers that display live database information in an InternetExpress application.

Class Hierarchy



Pascal

```
TCustomInetXPageProducer = class(TBaseInetXPageProducer, IWebContent);
```

C++

```
class TCustomInetXPageProducer : public TBaseInetXPageProducer, public IWebContent;
```

File

MidProd

Description

Use TCustomInetXPageProducer as a base class for components that create Web pages in an InternetExpress Web application. These pages display database information from an application server and include data packets encoded in XML and javascript that lets Web browsers navigate that data, post changes, and apply updates back to the application server.

The InetX page producer creates its HTML document from a template. The InetX page producer has a default template that coordinates several top-level declarations for the HTML document with content produced by other components. The other components that generate HTML content for the InetX page producer can be added at design time using the Web Page editor.

Do not instantiate TCustomInetXPageProducer. To create a producer that generates HTML pages with javascript database functions and XML data packets, use a descendant class such as TInetXPageProducer (see page 2285).

See Also

TXMLBroker

38.1.2.1 TCustomInetXPageProducer Members

The following tables list the members exposed by TCustomInetXPageProducer.

Protected Methods

	Name	Description
	AddIncludes (see page 2261)	This is AddIncludes, a member of class TBaseInetXPageProducer.
	AddScriptComponents (see page 2261)	This is AddScriptComponents, a member of class TBaseInetXPageProducer.
	CanAddClass (see page 2261)	Indicates whether a specified class can be used as a child Web item.
	CreatePageElements (see page 2262)	This is CreatePageElements, a member of class TBaseInetXPageProducer.
	CreateScriptManager (see page 2262)	This is CreateScriptManager, a member of class TBaseInetXPageProducer.
	DeclareFunctions (see page 2263)	This is DeclareFunctions, a member of class TBaseInetXPageProducer.
	DeclareIncludes (see page 2263)	This is DeclareIncludes, a member of class TBaseInetXPageProducer.
	DeclareRowSets (see page 2263)	This is DeclareRowSets, a member of class TBaseInetXPageProducer.
	DeclareScript (see page 2263)	This is DeclareScript, a member of class TBaseInetXPageProducer.
	DeclareStyles (see page 2263)	This is DeclareStyles, a member of class TBaseInetXPageProducer.
	DeclareWarnings (see page 2264)	This is DeclareWarnings, a member of class TBaseInetXPageProducer.

	DeclareXMLDocuments (see page 2264)	This is DeclareXMLDocuments, a member of class TBaseInetXPageProducer.
	DeclareXMLIslands (see page 2264)	This is DeclareXMLIslands, a member of class TBaseInetXPageProducer.
	DoAfterGetContent (see page 2264)	This is DoAfterGetContent, a member of class TBaseInetXPageProducer.
	DoAfterGetXMLData (see page 2265)	This is DoAfterGetXMLData, a member of class TBaseInetXPageProducer.
	DoBeforeGetContent (see page 2265)	This is DoBeforeGetContent, a member of class TBaseInetXPageProducer.
	DoBeforeGetXMLData (see page 2265)	This is DoBeforeGetXMLData, a member of class TBaseInetXPageProducer.
	DoTagEvent (see page 2265)	This is DoTagEvent, a member of class TBaseInetXPageProducer.
	GeneratePageElements (see page 2266)	This is GeneratePageElements, a member of class TBaseInetXPageProducer.
	GetChildren (see page 2968)	Enumerates all child components.
	GetComponentList (see page 2968)	This is GetComponentList, a member of class TBasePageItemsProducer.
	GetContentOptions (see page 2267)	This is GetContentOptions, a member of class TBaseInetXPageProducer.
	GetDefaultComponentList (see page 2969)	This is GetDefaultComponentList, a member of class TBasePageItemsProducer.
	GetDefaultTemplate (see page 2267)	This is GetDefaultTemplate, a member of class TBaseInetXPageProducer.
	GetScriptManager (see page 2268)	This is GetScriptManager, a member of class TBaseInetXPageProducer.
	GetTemplateFileName (see page 2280)	This is GetTemplateFileName, a member of class TCustomInetXPageProducer.
	GetTemplateStream (see page 2280)	This is GetTemplateStream, a member of class TCustomInetXPageProducer.
	GetUseXMLIslands (see page 2268)	This is GetUseXMLIslands, a member of class TBaseInetXPageProducer.
	GetXMLDelta (see page 2268)	This is GetXMLDelta, a member of class TBaseInetXPageProducer.
	GetXMLErrors (see page 2269)	This is GetXMLErrors, a member of class TBaseInetXPageProducer.
	ImplCanAddClass (see page 2269)	This is ImplCanAddClass, a member of class TBaseInetXPageProducer.
	SetChildOrder (see page 2969)	Sets the visual ordering of a child control.
	SetWebContentOptions (see page 2270)	This is SetWebContentOptions, a member of class TBaseInetXPageProducer.
	WebContent (see page 2281)	This is WebContent, a member of class TCustomInetXPageProducer.

Protected Properties

	Name	Description
	PageElements (see page 2271)	This is PageElements, a member of class TBaseInetXPageProducer.
	ScriptManager (see page 2272)	This is ScriptManager, a member of class TBaseInetXPageProducer.

Public Events

	Name	Description
	OnAfterGetContent (see page 2273)	Occurs after the page producer generates its HTML document.
	OnAfterGetXMLData (see page 2274)	Occurs after the page producer translates fetches an XML data packet from an XML broker.
	OnBeforeGetContent (see page 2274)	Occurs when the Content method is called.
	OnBeforeGetXMLData (see page 2275)	Occurs before the page producer fetches an XML data packet from an XML broker.

Public Methods

	Name	Description
	Content (see page 2279)	Returns the HTML document assembled by the InternetExpress page producer.
	ContentFromStream (see page 2262)	Returns the content produced in response to input read from a stream.
	Create (see page 2278)	Creates and initializes an instance of TCustomInetXPageProducer (see page 2275).
	Destroy (see page 2278)	Frees an instance of TCustomInetXPageProducer (see page 2275).
	FindXMLBroker (see page 2265)	Returns the XML broker that supplies the data packet specified by a DATAPACKET tag.
	FindXMLBrokerName (see page 2266)	Returns the name of an XML broker given the parameters in a DATAPACKET tag..
	GetRequiredTags (see page 2267)	Returns the default set of HTML-transparent tags for an InternetExpress page producer.
	GetXMLData (see page 2268)	Obtains an XML data packet from a specified XML broker.
	SetIncludePathURL (see page 2269)	Assigns a value to the IncludePathURL (see page 2271) property.
	SetStyles (see page 2269)	Assigns a value to the Styles (see page 2272) property.
	SetStylesFile (see page 2270)	Assigns a value to the StylesFile (see page 2273) property.

Public Properties

	Name	Description
	EnableXMLIslands (see page 2271)	Specifies whether XML data packets should be declared using XML islands if the requesting browser supports them.
	HTMLDoc (see page 2281)	Specifies the HTML template used to generate the value returned by the Content (see page 2279) method.
	HTMLFile (see page 2282)	Specifies the name of a file that contains the HTML template used to generate the value returned by the Content (see page 2279) method.
	IncludePathURL (see page 2271)	Indicates the URL where the javascript libraries are installed.
	Styles (see page 2272)	Defines the styles that can be used by the page producer and its Web items.
	StylesFile (see page 2273)	Specifies a file containing the individual styles for the page producer's style sheet.
R	WebPageItems (see page 2969)	Maintains a list of components that generate HTML content for the producer.

38.1.2.2 TCustomInetXPageProducer.Create Constructor

Creates and initializes an instance of TCustomInetXPageProducer (see page 2275).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomInetXPageProducer(TComponent * AOwner);
```

Description

Call Create to instantiate an InternetExpress page producer at runtime. InternetExpress page producers placed in a Web module at design time are instantiated automatically.

AOwner is a component, typically a Web module, that becomes the value of the Owner property. The Owner is responsible for freeing the TCustomInetXPageProducer (see page 2275) instance.

See Also

Owner

38.1.2.3 TCustomInetXPageProducer.Destroy Destructor

Frees an instance of TCustomInetXPageProducer (see page 2275).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomInetXPageProducer();
```

Description

Do not call Destroy directly. Instead, call the Free method, which checks whether the reference to the producer is not nil before calling Destroy.

See Also

System::TObject::Free

38.1.2.4 TCustomInetXPageProducer Methods

The methods of the TCustomInetXPageProducer class are listed here.

Protected Methods

	Name	Description
	GetTemplateFileName (see page 2280)	This is GetTemplateFileName, a member of class TCustomInetXPageProducer.
	GetTemplateStream (see page 2280)	This is GetTemplateStream, a member of class TCustomInetXPageProducer.
	WebContent (see page 2281)	This is WebContent, a member of class TCustomInetXPageProducer.

Public Methods

	Name	Description
	Content (see page 2279)	Returns the HTML document assembled by the InternetExpress page producer.

38.1.2.4.1 TCustomInetXPageProducer.Content Method

Returns the HTML document assembled by the InternetExpress page producer.

Pascal

```
function Content: string; override;
```

C++

```
virtual __fastcall UnicodeString Content();
```

Description

Call Content to get the HTML document produced by TCustomInetXPageProducer (see page 2275). This document contains a mixture of HTML, XML, and javascript. It allows Web browsers to view and edit data from an application server.

The InternetExpress page producer creates its content by replacing HTML-transparent tags in the template specified by HTMLDoc (see page 2281) or HTMLFile (see page 2282).

By default, this template has the form

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<#INCLUDES> <#STYLES> <#WARNINGS> <#FORMS> <#SCRIPT>
</BODY>
</HTML>
```

Although the template can be changed to include any combination of these tags with tags of the following forms

```
<#COMPONENT Name=WebItemName>
<#DATAPACKET XMLBroker=BrokerName>
```

which are also translated automatically. The template can replace the default HTML-transparent tags with the tag

```
<#BODYELEMENTS>
```

Finally, the template can include any tags translated by the OnHTMLTag event handler.

The Content method translates the tags listed above as follows:

<#INCLUDES> generates the statements that include the javascript libraries at the location specified by IncludePathURL (see page 2271).

<#STYLES> generates the statements that defines a style sheet from the definitions listed in the Styles (see page 2272) or StylesFile (see page 2273) property.

<#WARNINGS> generates nothing at runtime. At design time, it adds warning messages for problems detected while generating the HTML document. You can see these messages in the Web page editor.

<#FORMS> generates the HTML produced by the components listed in the WebPageItems property. The HTML from each component is listed in the order it appears in WebPageItems.

<#SCRIPT> generates a block of javascript declarations that are used by the components listed in WebPageItems.

<#BODYELEMENTS> generates the same HTML as results from the 5 tags in the default template. It is useful when generating a template in an HTML editor when you want to use the default layout but add additional elements (such as style sheets) using the editor.

<#COMPONENT> is replaced with the content produced by the specified component. Typically, this is a component listed in the WebPageItems property, but it can be any component that supports the IWebContent interface and has the same Owner as the INETX page producer.

<#DATAPACKET> is replaced with the XML data packet obtained from the specified XML broker.

The string returned by Content can be used as the Content property of the response to an HTTP request message.

See Also

IWebContent

IncludePathURL (see page 2271)

Styles (see page 2272)

StylesFile (see page 2273)

OnHTMLTag

HTMLDoc (see page 2281)

HTMLFile (see page 2282)

WebPageItems (see page 2969)

Customizing the InternetExpress Page Producer Template

38.1.2.4.2 TCustomInetXPageProducer.GetTemplateFileName Method

Pascal

```
function GetTemplateFileName: string; override;
```

C++

```
virtual __fastcall UnicodeString GetTemplateFileName();
```

Description

This is GetTemplateFileName, a member of class TCustomInetXPageProducer.

38.1.2.4.3 TCustomInetXPageProducer.GetTemplateStream Method

Pascal

```
function GetTemplateStream(out AOwned: Boolean): TStream; override;
```

C++

```
virtual __fastcall TStream GetTemplateStream(Boolean AOwned);
```

Description

This is GetTemplateStream, a member of class TCustomInetXPageProducer.

38.1.2.4.4 TCustomInetXPageProducer.WebContent Method

Pascal

```
function WebContent(Options: TWebContentOptions; Layout: TLayout): string;
```

C++

```
__fastcall UnicodeString WebContent(TWebContentOptions Options, TLayout Layout);
```

Description

This is WebContent, a member of class TCustomInetXPageProducer.

38.1.2.5 TCustomInetXPageProducer Properties

The properties of the TCustomInetXPageProducer class are listed here.

Public Properties

	Name	Description
	HTMLDoc (see page 2281)	Specifies the HTML template used to generate the value returned by the Content (see page 2279) method.
	HTMLFile (see page 2282)	Specifies the name of a file that contains the HTML template used to generate the value returned by the Content (see page 2279) method.

38.1.2.5.1 TCustomInetXPageProducer.HTMLDoc Property

Specifies the HTML template used to generate the value returned by the Content (see page 2279) method.

Pascal

```
property HTMLDoc: TStrings;
```

C++

```
__property TStrings HTMLDoc;
```

Description

Set HTMLDoc to a set of strings that make up the HTML template. By default, HTMLDoc is initialized to the following template:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<#INCLUDES> <#STYLES> <#WARNINGS> <#FORMS> <#SCRIPT>
</BODY>
</HTML>
```

Each string is a sequence of one or more HTML commands or HTML-transparent tags. An HTML-transparent tag has the form

```
<#TagName Param1=Value1 Param2=Value2 ...>
```

A pound sign (#) immediately follows the opening angle bracket (<) with no spaces separating it from the angle bracket. The pound sign identifies the string to the page producer as an HTML-transparent tag. The tag name immediately follows the pound sign with no spaces separating it from the pound sign. Following the tag name, the HTML-transparent tag can optionally include parameters that specify details of the conversion to be performed. Each parameter is of the form ParamName=Value, where there is no space between the parameter name, the equals symbol (=) and the value. Each parameter is delimited by white

space.

The angle brackets (< >) make the tag transparent to HTML browsers that do not recognize the #TagName construct.

The page producer converts the entire HTML-transparent tag into true HTML commands based on the value of the tag name. TCustomInetXPageProducer (see page 2275) automatically handles the conversion of the HTML-transparent tags in the default template, as well as the tags described in the help for the Content (see page 2279) method. If you set HTMLDoc to include other HTML-transparent tags, you must translate these in an OnHTMLTag event handler.

Note: Setting HTMLDoc clears the HTMLFile (see page 2282) property.

See Also

Content (see page 2279)

HTMLFile (see page 2282)

OnHTMLTag

Creating Web Pages with an InternetExpress Page Producer

Customizing the InternetExpress Page Producer Template

38.1.2.5.2 TCustomInetXPageProducer.HTMLFile Property

Specifies the name of a file that contains the HTML template used to generate the value returned by the Content (see page 2279) method.

Pascal

```
property HTMLFile: TFileName;
```

C++

```
__property TFileName HTMLFile;
```

Description

Set HTMLFile to the name of the file that contains an HTML template. The HTML template is a sequence of HTML commands and HTML-transparent tags. An HTML-transparent tag has the form

```
<#TagName Param1=Value1 Param2=Value2 ...>
```

A pound sign (#) immediately follows the opening angle bracket (<) with no spaces separating it from the angle bracket. The pound sign identifies the string to the page producer as an HTML-transparent tag. The tag name immediately follows the pound sign with no spaces separating it from the pound sign. The page producer converts the entire HTML-transparent tag into true HTML commands based on the value of the tag name. Following the tag name, the HTML-transparent tag can optionally include parameters that specify details of the conversion to be performed. Each parameter is of the form ParamName=Value, where there is no space between the parameter name, the equals symbol (=) and the value. Each parameter is delimited by white space.

The angle brackets (< >) make the tag transparent to HTML browsers that do not recognize the #TagName construct.

Note: Setting HTMLFile clears the HTMLDoc (see page 2281) property.

Note: When distributing applications that contain page producers which use the HTMLFile property, be sure to distribute the file along with the application. Also, be sure that the value of HTMLFile does not include a hard-coded path name that will not be available on the user's disk.

See Also

Content (see page 2279)

HTMLDoc (see page 2281)

Customizing the InternetExpress Page Producer Template

38.1.3 TInetXPageElements Class

Class Hierarchy

Pascal

```
TInetXPageElements = class(TObject);
```

C++

```
class TInetXPageElements : public TObject;
```

File

MidProd

Description

This is class MidProd.TInetXPageElements.

38.1.3.1 TInetXPageElements Members

The following tables list the members exposed by TInetXPageElements.

Public Fields

	Name	Description
	FormsContent (see page 2284)	This is FormsContent, a member of class TInetXPageElements.
	IncludesContent (see page 2284)	This is IncludesContent, a member of class TInetXPageElements.
	ScriptContent (see page 2284)	This is ScriptContent, a member of class TInetXPageElements.
	StylesContent (see page 2284)	This is StylesContent, a member of class TInetXPageElements.
	WarningsContent (see page 2284)	This is WarningsContent, a member of class TInetXPageElements.

Public Methods

	Name	Description
	BodyContent (see page 2285)	This is BodyContent, a member of class TInetXPageElements.

38.1.3.2 TInetXPageElements Fields

The fields of the TInetXPageElements class are listed here.

Public Fields

	Name	Description
	FormsContent (see page 2284)	This is FormsContent, a member of class TInetXPageElements.
	IncludesContent (see page 2284)	This is IncludesContent, a member of class TInetXPageElements.
	ScriptContent (see page 2284)	This is ScriptContent, a member of class TInetXPageElements.
	StylesContent (see page 2284)	This is StylesContent, a member of class TInetXPageElements.
	WarningsContent (see page 2284)	This is WarningsContent, a member of class TInetXPageElements.

38.1.3.2.1 TInetXPageElements.FormsContent Field

Pascal

```
FormsContent: string;
```

C++

```
UnicodeString FormsContent;
```

Description

This is FormsContent, a member of class TInetXPageElements.

38.1.3.2.2 TInetXPageElements.IncludesContent Field

Pascal

```
IncludesContent: string;
```

C++

```
UnicodeString IncludesContent;
```

Description

This is IncludesContent, a member of class TInetXPageElements.

38.1.3.2.3 TInetXPageElements.ScriptContent Field

Pascal

```
ScriptContent: string;
```

C++

```
UnicodeString ScriptContent;
```

Description

This is ScriptContent, a member of class TInetXPageElements.

38.1.3.2.4 TInetXPageElements.StylesContent Field

Pascal

```
StylesContent: string;
```

C++

```
UnicodeString StylesContent;
```

Description

This is StylesContent, a member of class TInetXPageElements.

38.1.3.2.5 TInetXPageElements.WarningsContent Field

Pascal

```
WarningsContent: string;
```

C++

```
UnicodeString WarningsContent;
```

Description

This is WarningsContent, a member of class TInetXPageElements.

38.1.3.3 TInetXPageElements Methods

The methods of the TInetXPageElements class are listed here.

Public Methods

	Name	Description
	BodyContent (see page 2285)	This is BodyContent, a member of class TInetXPageElements.

38.1.3.3.1 TInetXPageElements.BodyContent Method

Pascal

```
function BodyContent: string; virtual;
```

C++

```
virtual __fastcall UnicodeString BodyContent();
```

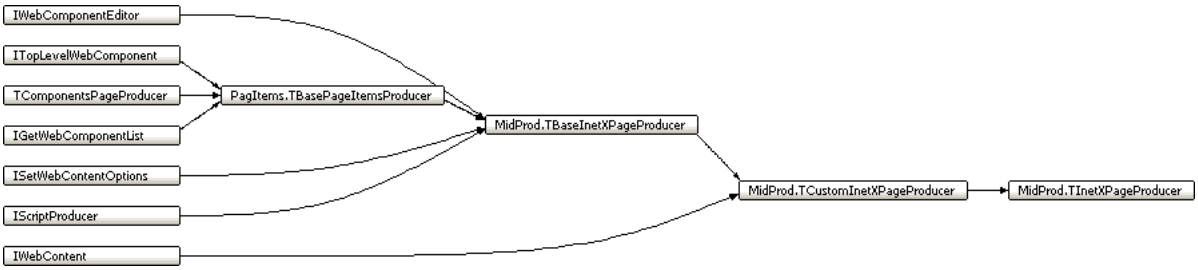
Description

This is BodyContent, a member of class TInetXPageElements.

38.1.4 TInetXPageProducer Class

TInetXPageProducer assembles an HTML document that represents live database information from an application server.

Class Hierarchy



Pascal

```
TInetXPageProducer = class(TCustomInetXPageProducer);
```

C++

```
class TInetXPageProducer : public TCustomInetXPageProducer;
```

File

MidProd

Description

Use TInetXPageProducer to create Web pages for an InternetExpress application. These pages display database information from an application server and include data packets encoded in XML and javascript that lets Web browsers navigate that data,

post changes, and apply updates back to the application server.

The InetX page producer is assigned as the Producer property of a Web action item or called from an OnAction event handler.

The InetX page producer creates its HTML document from a template. Unlike other page producers, the InetX page producer has a default template that coordinates several top-level declarations for the HTML document with content produced by other components. The other components that generate HTML content for the InetX page producer can be added at design time using the Web Page editor.

See Also

TWebActionItem

TXMLBroker

The Web Module

38.1.4.1 TInetXPageProducer Members

The following tables list the members exposed by TInetXPageProducer.

Protected Methods

	Name	Description
	AddIncludes (see page 2261)	This is AddIncludes, a member of class TBaseInetXPageProducer.
	AddScriptComponents (see page 2261)	This is AddScriptComponents, a member of class TBaseInetXPageProducer.
	CanAddClass (see page 2261)	Indicates whether a specified class can be used as a child Web item.
	CreatePageElements (see page 2262)	This is CreatePageElements, a member of class TBaseInetXPageProducer.
	CreateScriptManager (see page 2262)	This is CreateScriptManager, a member of class TBaseInetXPageProducer.
	DeclareFunctions (see page 2263)	This is DeclareFunctions, a member of class TBaseInetXPageProducer.
	DeclareIncludes (see page 2263)	This is DeclareIncludes, a member of class TBaseInetXPageProducer.
	DeclareRowSets (see page 2263)	This is DeclareRowSets, a member of class TBaseInetXPageProducer.
	DeclareScript (see page 2263)	This is DeclareScript, a member of class TBaseInetXPageProducer.
	DeclareStyles (see page 2263)	This is DeclareStyles, a member of class TBaseInetXPageProducer.
	DeclareWarnings (see page 2264)	This is DeclareWarnings, a member of class TBaseInetXPageProducer.
	DeclareXMLDocuments (see page 2264)	This is DeclareXMLDocuments, a member of class TBaseInetXPageProducer.
	DeclareXMLIslands (see page 2264)	This is DeclareXMLIslands, a member of class TBaseInetXPageProducer.
	DoAfterGetContent (see page 2264)	This is DoAfterGetContent, a member of class TBaseInetXPageProducer.
	DoAfterGetXMLData (see page 2265)	This is DoAfterGetXMLData, a member of class TBaseInetXPageProducer.
	DoBeforeGetContent (see page 2265)	This is DoBeforeGetContent, a member of class TBaseInetXPageProducer.
	DoBeforeGetXMLData (see page 2265)	This is DoBeforeGetXMLData, a member of class TBaseInetXPageProducer.
	DoTagEvent (see page 2265)	This is DoTagEvent, a member of class TBaseInetXPageProducer.
	GeneratePageElements (see page 2266)	This is GeneratePageElements, a member of class TBaseInetXPageProducer.
	GetChildren (see page 2968)	Enumerates all child components.
	GetComponentList (see page 2968)	This is GetComponentList, a member of class TBasePageItemsProducer.
	GetContentOptions (see page 2267)	This is GetContentOptions, a member of class TBaseInetXPageProducer.
	GetDefaultComponentList (see page 2969)	This is GetDefaultComponentList, a member of class TBasePageItemsProducer.
	GetDefaultTemplate (see page 2267)	This is GetDefaultTemplate, a member of class TBaseInetXPageProducer.
	GetScriptManager (see page 2268)	This is GetScriptManager, a member of class TBaseInetXPageProducer.
	GetTemplateFileName (see page 2280)	This is GetTemplateFileName, a member of class TCustomInetXPageProducer.
	GetTemplateStream (see page 2280)	This is GetTemplateStream, a member of class TCustomInetXPageProducer.
	GetUseXMLIslands (see page 2268)	This is GetUseXMLIslands, a member of class TBaseInetXPageProducer.
	GetXMLDelta (see page 2268)	This is GetXMLDelta, a member of class TBaseInetXPageProducer.
	GetXMLErrors (see page 2269)	This is GetXMLErrors, a member of class TBaseInetXPageProducer.
	ImplCanAddClass (see page 2269)	This is ImplCanAddClass, a member of class TBaseInetXPageProducer.
	SetChildOrder (see page 2969)	Sets the visual ordering of a child control.
	SetWebContentOptions (see page 2270)	This is SetWebContentOptions, a member of class TBaseInetXPageProducer.

	WebContent (see page 2281)	This is WebContent, a member of class TCustomInetXPageProducer.

Protected Properties

	Name	Description
R	PageElements (see page 2271)	This is PageElements, a member of class TBaseInetXPageProducer.
R	ScriptManager (see page 2272)	This is ScriptManager, a member of class TBaseInetXPageProducer.

Public Methods

	Name	Description
V	Content (see page 2279)	Returns the HTML document assembled by the InternetExpress page producer.
V	ContentFromStream (see page 2262)	Returns the content produced in response to input read from a stream.
V	Create (see page 2259)	Creates an instance of a TBaseInetXPageProducer (see page 2257).
V	Create (see page 2278)	Creates and initializes an instance of TCustomInetXPageProducer (see page 2275).
V	Create (see page 2967)	Creates an instance of a TBasePageItemsProducer (see page 2966).
V	Destroy (see page 2259)	Destroys the instance of the TBasePageItemsProducer.
V	Destroy (see page 2278)	Frees an instance of TCustomInetXPageProducer (see page 2275).
V	Destroy (see page 2967)	Destroys the instance of the TBasePageItemsProducer (see page 2966).
V	FindXMLBroker (see page 2265)	Returns the XML broker that supplies the data packet specified by a DATAPACKET tag.
V	FindXMLBrokerName (see page 2266)	Returns the name of an XML broker given the parameters in a DATAPACKET tag..
V	GetRequiredTags (see page 2267)	Returns the default set of HTML-transparent tags for an InternetExpress page producer.
V	GetXMLData (see page 2268)	Obtains an XML data packet from a specified XML broker.
	SetIncludePathURL (see page 2269)	Assigns a value to the IncludePathURL (see page 2271) property.
	SetStyles (see page 2269)	Assigns a value to the Styles (see page 2272) property.
	SetStylesFile (see page 2270)	Assigns a value to the StylesFile (see page 2273) property.

Published Events

	Name	Description
	OnAfterGetContent (see page 2293)	Occurs after the page producer generates its HTML document.
	OnAfterGetXMLData (see page 2293)	Occurs after the page producer translates fetches an XML data packet from an XML broker.
	OnBeforeGetContent (see page 2294)	Occurs when the Content method is called.
	OnBeforeGetXMLData (see page 2294)	Occurs before the page producer fetches an XML data packet from an XML broker.

Published Properties

	Name	Description
	EnableXMLIslands (see page 2288)	Specifies whether XML data packets should be declared using XML islands if the requesting browser supports them.
	HTMLDoc (see page 2288)	Specifies the HTML template used to generate the value returned by the Content method.
	HTMLFile (see page 2289)	Specifies the name of a file that contains the HTML template used to generate the value returned by the Content method.
	IncludePathURL (see page 2290)	Indicates the URL where the javascript libraries are installed.
	ScriptEngine (see page 2290)	Names the script engine that generates or interprets script embedded in the documents that this producer creates.
	Styles (see page 2291)	Defines the styles that can be used by the page producer and its Web items.
	StylesFile (see page 2291)	Specifies a file containing the individual styles for the page producer's style sheet.
	WebPageItems (see page 2292)	Maintains a list of components that generate HTML content for the producer.

38.1.4.2 TInetXPageProducer Properties

The properties of the TInetXPageProducer class are listed here.

Published Properties

	Name	Description
	EnableXMLIslands (see page 2288)	Specifies whether XML data packets should be declared using XML islands if the requesting browser supports them.
	HTMLDoc (see page 2288)	Specifies the HTML template used to generate the value returned by the Content method.
	HTMLFile (see page 2289)	Specifies the name of a file that contains the HTML template used to generate the value returned by the Content method.
	IncludePathURL (see page 2290)	Indicates the URL where the javascript libraries are installed.
	ScriptEngine (see page 2290)	Names the script engine that generates or interprets script embedded in the documents that this producer creates.
	Styles (see page 2291)	Defines the styles that can be used by the page producer and its Web items.
	StylesFile (see page 2291)	Specifies a file containing the individual styles for the page producer's style sheet.
	WebPageItems (see page 2292)	Maintains a list of components that generate HTML content for the producer.

38.1.4.2.1 TInetXPageProducer.EnableXMLIslands Property

Specifies whether XML data packets should be declared using XML islands if the requesting browser supports them.

Pascal

```
property EnableXMLIslands: Boolean;
```

C++

```
__property Boolean EnableXMLIslands;
```

Description

Set EnableXMLIslands to indicate whether the XML in the generated HTML document should be declared using XML islands. XML islands are only used if the document is created in response to a request message whose headers indicate that the browser supports XML islands.

Set EnableXMLIslands to false to prevent the page producer from declaring XML islands even if the requesting browser supports them.

See Also

GetXMLData (see page 2268)

Content

38.1.4.2.2 TInetXPageProducer.HTMLDoc Property

Specifies the HTML template used to generate the value returned by the Content method.

Pascal

```
property HTMLDoc: TStrings;
```

C++

```
__property TStrings HTMLDoc;
```

Description

Set HTMLDoc to a set of strings that make up the HTML template. By default, HTMLDoc is initialized to the following template:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<#INCLUDES> <#STYLES> <#WARNINGS> <#FORMS> <#SCRIPT>
</BODY>
```

```
</HTML>
```

Each string is a sequence of one or more HTML commands or HTML-transparent tags. An HTML-transparent tag has the form

```
<#TagName Param1=Value1 Param2=Value2 ...>
```

A pound sign (#) immediately follows the opening angle bracket (<) with no spaces separating it from the angle bracket. The pound sign identifies the string to the page producer as an HTML-transparent tag. The tag name immediately follows the pound sign with no spaces separating it from the pound sign. Following the tag name, the HTML-transparent tag can optionally include parameters that specify details of the conversion to be performed. Each parameter is of the form ParamName=Value, where there is no space between the parameter name, the equals symbol (=) and the value. Each parameter is delimited by white space.

The angle brackets (< >) make the tag transparent to HTML browsers that do not recognize the #TagName construct.

The page producer converts the entire HTML-transparent tag into true HTML commands based on the value of the tag name. TCustomInetXPageProducer automatically handles the conversion of the HTML-transparent tags in the default template, as well as the tags described in the help for the Content method. If you set HTMLDoc to include other HTML-transparent tags, you must translate these in an OnHTMLTag event handler.

Note: Setting HTMLDoc clears the HTMLFile property.

See Also

Content (see page 2279)

HTMLFile (see page 2282)

OnHTMLTag

Creating Web Pages with an InternetExpress Page Producer

Customizing the InternetExpress Page Producer Template

38.1.4.2.3 TInetXPageProducer.HTMLFile Property

Specifies the name of a file that contains the HTML template used to generate the value returned by the Content method.

Pascal

```
property HTMLFile: TFileName;
```

C++

```
__property TFileName HTMLFile;
```

Description

Set HTMLFile to the name of the file that contains an HTML template. The HTML template is a sequence of HTML commands and HTML-transparent tags. An HTML-transparent tag has the form

```
<#TagName Param1=Value1 Param2=Value2 ...>
```

A pound sign (#) immediately follows the opening angle bracket (<) with no spaces separating it from the angle bracket. The pound sign identifies the string to the page producer as an HTML-transparent tag. The tag name immediately follows the pound sign with no spaces separating it from the pound sign. The page producer converts the entire HTML-transparent tag into true HTML commands based on the value of the tag name. Following the tag name, the HTML-transparent tag can optionally include parameters that specify details of the conversion to be performed. Each parameter is of the form ParamName=Value, where there is no space between the parameter name, the equals symbol (=) and the value. Each parameter is delimited by white space.

The angle brackets (< >) make the tag transparent to HTML browsers that do not recognize the #TagName construct.

Note: Setting HTMLFile clears the HTMLDoc property.

Note: When distributing applications that contain page producers which use the HTMLFile property, be sure to distribute the file along with the application. Also, be sure that the value of HTMLFile does not include a hard-coded path name that will not be available on the user's disk.

See Also

Content (see page 2279)

HTMLDoc (see page 2281)

Customizing the InternetExpress Page Producer Template

38.1.4.2.4 TInetXPageProducer.IncludePathURL Property

Indicates the URL where the javascript libraries are installed.

Pascal

```
property IncludePathURL: string;
```

C++

```
__property UnicodeString IncludePathURL;
```

Description

Set IncludePathURL to indicate where the javascript libraries are installed. The page producer uses this value to create HTML statements that include the libraries in the generated HTML document. These statements have the form

```
<SCRIPT language=Javascript type="text/javascript" SRC="IncludePathURL/xmldb.js"> </SCRIPT>
```

The javascript libraries define classes and methods that allow the HTML document to interpret XML data packets if the Web browser is not XML-enabled, and that enable controls in the document to navigate the data, post changes, and apply updates.

See Also

SetIncludePathURL (see page 2269)

Content

Creating Web Pages with an InternetExpress Page Producer

38.1.4.2.5 TInetXPageProducer.ScriptEngine Property

Names the script engine that generates or interprets script embedded in the documents that this producer creates.

Pascal

```
property ScriptEngine;
```

C++

```
__property ScriptEngine;
```

Description

ScriptEngine indicates the scripting language that the producer uses in its generated documents. For example, if the producer generates code that includes jscript, the value of ScriptEngine is 'JSCRIPT'.

When ScriptEngine is defined, the page producer generates its content using a special script-producer component. When ScriptEngine is not defined, the page producer uses only the OnHTMLTag event, plus any other built-in translations provided by a descendant class.

See Also

OnHTMLTag

38.1.4.2.6 TInetXPageProducer.Styles Property

Defines the styles that can be used by the page producer and its Web items.

Pascal

```
property Styles: TStrings;
```

C++

```
__property TStrings Styles;
```

Description

Use Styles to define a style sheet for the HTML document produced by TBaseInetXPageProducer. Each string in the TStrings object specifies a specific style by giving a style selector followed by a set of attributes in curly braces. These definitions can define styles for standard HTML elements such as

```
H2 B  {color: red}
```

or they can define styles that you name, such as

```
.MyStyle  {font-family: arial; font-weight: bold; font-size: 18px }
```

Note that user-defined style names must begin with a dot.

Styles for standard elements are applied automatically to HTML that uses the tag or tags that form the style selector. User-defined styles can be applied by setting the StyleRule property of a Web item to the user-defined style name that forms the selector.

Note: If you are sharing a set of styles between multiple page producers or multiple applications, you can define the styles in a file. To define the styles in a file, use the StylesFile property instead. If StylesFile is defined, it takes precedence over the Styles property.

See Also

SetStyles (see page 2269)

StylesFile (see page 2273)

Content

Customizing the InternetExpress Page Producer Template

Setting Web Item Properties

38.1.4.2.7 TInetXPageProducer.StylesFile Property

Specifies a file containing the individual styles for the page producer's style sheet.

Pascal

```
property StylesFile: TFileName;
```

C++

```
__property TFileName StylesFile;
```

Description

Set StylesFile to the name of a file containing the style definitions used by the page producer. The page producer uses these

style definitions to declare a style sheet in the HTML document it generates.

Styles for standard elements are applied automatically to HTML in the document. User-defined styles can be applied by setting the StyleRule property of a Web item to the user-defined style name that forms the selector of a style definition.

Note: If the StylesFile property is not set, you can specify the style definitions using the Styles property instead.

See Also

SetStylesFile (see page 2270)

Styles (see page 2272)

Content

Customizing the InternetExpress Page Producer Template

Setting Web Item Properties

38.1.4.2.8 TInetXPageProducer.WebPageItems Property

Maintains a list of components that generate HTML content for the producer.

Pascal

```
property WebPageItems: TWebComponentList;
```

C++

```
__property TWebComponentList WebPageItems;
```

Description

WebPageItems lists the components that generate content for the page items producer. Each of these components supports the IWebContent interface, and can contain its own list of Web items that help it generate its content.

The type of items that can be added to the WebPageItems list are restricted by a protected method that is abstract or, in C++ terminology, pure virtual in TBasePageItemsProducer. Descendant classes implement this protected method to indicate what types of items can be added to the list.

At design time, use the Web page editor to add components to the WebPageItems property.

Note: Typically, WebPageItems lists all the components that generate content for the page items producer, in the order that the generated content appears in the final HTML document. However, the page items producer can transform COMPONENT tags in its template using components that are not in this list, as long as they have the same owner as the page items producer..

See Also

IWebContent

Creating Web Pages with an InternetExpress Page Producer

38.1.4.3 TInetXPageProducer Events

The events of the TInetXPageProducer class are listed here.

Published Events

	Name	Description
	OnAfterGetContent (see page 2293)	Occurs after the page producer generates its HTML document.
	OnAfterGetXMLData (see page 2293)	Occurs after the page producer translates fetches an XML data packet from an XML broker.
	OnBeforeGetContent (see page 2294)	Occurs when the Content method is called.

	OnBeforeGetXMLData (see page 2294)	Occurs before the page producer fetches an XML data packet from an XML broker.

38.1.4.3.1 TInetXPageProducer.OnAfterGetContent Event

Occurs after the page producer generates its HTML document.

Pascal

```
property OnAfterGetContent: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnAfterGetContent;
```

Description

Write an OnAfterGetContent event handler to take specific action immediately after the page producer generates an HTML document. This event occurs immediately before the Content method returns the generated document. Typically it is used in applications where the page producer automatically responds to messages as the Provider property of an action item or the Response producer property of an XML broker.

See Also

OnBeforeGetContent (see page 2274)

Content

Producer

38.1.4.3.2 TInetXPageProducer.OnAfterGetXMLData Event

Occurs after the page producer translates fetches an XML data packet from an XML broker.

Pascal

```
property OnAfterGetXMLData: TXMLDataEvent;
```

C++

```
__property TXMLDataEvent OnAfterGetXMLData;
```

Description

Write an OnAfterGetXMLData event handler to take specific action immediately after the page producer obtains an XML data packet from an XML broker. Typically, the OnAfterGetXMLData event handler responds to any values returned by the provider on the application server as the OwnerData parameter.

When the page producer generates its HTML document (that is, when the Content method executes), it adds code to define the XML data packets manipulated in the document. To fetch these XML data packets, the page producer calls an XML broker's RequestRecords method. OnAfterGetXMLData occurs after the page producer has returned an XML data packet but before it is assigned as part of the emerging content.

See Also

OnBeforeGetXMLData (see page 2275)

GetXMLData (see page 2268)

OnHTMLTag

Content

38.1.4.3.3 TInetXPageProducer.OnBeforeGetContent Event

Occurs when the Content method is called.

Pascal

```
property OnBeforeGetContent: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnBeforeGetContent;
```

Description

Write an OnBeforeGetContent event handler to take specific action immediately before the page producer generates an HTML document. This event occurs when the Content method is called before the page producer starts assembling its HTML document. Typically it is used in applications where the page producer automatically responds to messages as the Provider property of an action item or the Response producer property of an XML broker.

See Also

OnAfterGetContent (see page 2273)

Content

Producer

38.1.4.3.4 TInetXPageProducer.OnBeforeGetXMLData Event

Occurs before the page producer fetches an XML data packet from an XML broker.

Pascal

```
property OnBeforeGetXMLData: TXMLDataEvent;
```

C++

```
__property TXMLDataEvent OnBeforeGetXMLData;
```

Description

Write an OnBeforeGetXMLData event handler to take specific action immediately before the page producer requests an XML data packet from an XML broker. Typically, the OnBeforeGetXMLData event handler encodes any values for the provider on the application server as the OwnerData parameter or initializes properties on the XML broker that affect the data packet.

When the page producer generates its HTML document (that is, when the Content method executes), it adds code to define the XML data packets manipulated in the document. To fetch these XML data packets, the page producer calls an XML broker's RequestRecords method. OnBeforeGetXMLData occurs immediately before the page producer requests an XML data packet. Any value assigned to the OwnerData parameter is passed to the XML broker's OnRequestRecords event handler, and from there to the provider's OnBeforeGetRecords event handler.

See Also

OnAfterGetXMLData (see page 2274)

OnRequestRecords

GetXMLData (see page 2268)

Content

OnHTMLTag

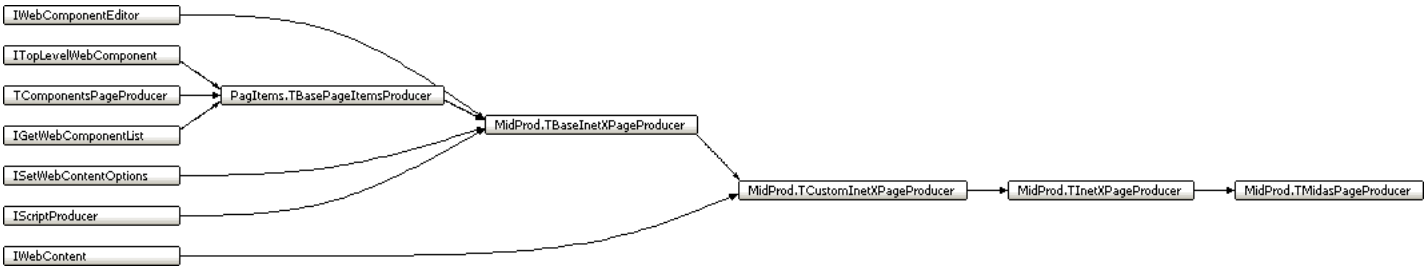
MaxRecords

Params

38.1.5 TMidasPageProducer Class

TMidasPageProducer assembles an HTML document that represents live database information from an application server.

Class Hierarchy



Pascal

```
TMidasPageProducer = class(TInetXPageProducer);
```

C++

```
class TMidasPageProducer : public TInetXPageProducer;
```

File

MidProd

Description

Do not use TMidasPageProducer. It is included only for backward compatibility. To add a page producer that assembles Web pages in an InternetExpress application, use TInetXPageProducer (see page 2285) instead.

See Also

TWebActionItem

TXMLBroker

38.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	AddStdIncludes (see page 2296)	This is function MidProd.AddStdIncludes.
	PathInfoToRelativePath (see page 2296)	This is function MidProd.PathInfoToRelativePath.

38.2.1 MidProd.AddStdIncludes Function

Pascal

```
procedure AddStdIncludes(StdIncludes: TStdIncludes; AddIntf: IAddScriptElements);
```

C++

```
AddStdIncludes(TStdIncludes StdIncludes, IAddScriptElements AddIntf);
```

File

MidProd

Description

This is function MidProd.AddStdIncludes.

38.2.2 MidProd.PathInfoToRelativePath Function

Pascal

```
function PathInfoToRelativePath(const PathInfo: string): string;
```

C++

```
UnicodeString PathInfoToRelativePath(const UnicodeString PathInfo);
```

File

MidProd

Description

This is function MidProd.PathInfoToRelativePath.

38.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TStdInclude (see page 2297)	This is record MidProd.TStdInclude.

38.3.1 MidProd.TStdInclude Enumeration

Pascal

```
TStdInclude = (
  jsDom,
  jsDB,
  jsDisp,
  jsBind,
  jsErrDisp,
  jsShow
);
```

C++

```
enum TStdInclude {
  jsDom,
  jsDB,
  jsDisp,
  jsBind,
  jsErrDisp,
  jsShow
};
```

File

MidProd

Description

This is record MidProd.TStdInclude.

38.4 Types

The following table lists types in this documentation.

Types

Name	Description
TBaseMidasPageProducer (see page 2298)	This is type MidProd.TBaseMidasPageProducer.
TCustomMidasPageProducer (see page 2298)	This is type MidProd.TCustomMidasPageProducer.
TMidasPageElements (see page 2299)	This is type MidProd.TMidasPageElements.
TStdIncludes (see page 2299)	This is type MidProd.TStdIncludes.
TXMLDataEvent (see page 2299)	TXMLDataEvent is the type of event handlers that occur when an InternetExpress page producer adds an XML data packet to its HTML document.

38.4.1 MidProd.TBaseMidasPageProducer Type

Pascal

```
TBaseMidasPageProducer = TBaseInetXPageProducer;
```

C++

```
TBaseInetXPageProducer TBaseMidasPageProducer;
```

File

MidProd

Description

This is type MidProd.TBaseMidasPageProducer.

38.4.2 MidProd.TCustomMidasPageProducer Type

Pascal

```
TCustomMidasPageProducer = TCustomInetXPageProducer;
```

C++

```
TCustomInetXPageProducer TCustomMidasPageProducer;
```

File

MidProd

Description

This is type MidProd.TCustomMidasPageProducer.

38.4.3 MidProd.TMidasPageElements Type

Pascal

```
TMidasPageElements = TInetXPageElements;
```

C++

```
TInetXPageElements TMidasPageElements;
```

File

MidProd

Description

This is type MidProd.TMidasPageElements.

38.4.4 MidProd.TStdIncludes Type

Pascal

```
TStdIncludes = set of TStdInclude;
```

C++

```
TStdInclude TStdIncludes;
```

File

MidProd

Description

This is type MidProd.TStdIncludes.

38.4.5 MidProd.TXMLDataEvent Type

TXMLDataEvent is the type of event handlers that occur when an InternetExpress page producer adds an XML data packet to its HTML document.

Pascal

```
TXMLDataEvent = procedure (Sender: TObject; Request: TWebRequest; XMLBroker: TXMLBroker; var
OwnerData: OleVariant) of object;
```

C++

```
(Sender: TObject; Request: TWebRequest; XMLBroker: TXMLBroker; var OwnerData: OleVariant) (
TXMLDataEvent)();
```

File

MidProd

Description

TXMLDataEvent is the type for event handlers that occur before and after an InternetExpress page producer translates a

DATAPACKET tag into a string of XML.

Sender is the InternetExpress page producer that is creating an HTML document containing the XML data packet.

Request is the request message in response to which the InternetExpress page producer is creating the HTML document.

XMLBroker is the XMLBroker that fetches the XML data packet from the application server and returns it to the InternetExpress page producer.

OwnerData is application-specific information that allows the InternetExpress page producer to communicate with the XMLBroker and with the provider on the application server. Typically, this parameter is used to communicate persistent state information when the application server is stateless. If the event handler is used for an OnBeforeGetXMLData event, the event handler creates and returns a value in OwnerData which is passed to the XMLBroker's OnRequestRecords event handler, and from there passed to the provider on the application server. If the event handler is used for an OnAfterGetXMLData event, OwnerData is set to the value returned by the provider on the application server on entry into the event handler and is ignored on exit.

See Also

TBaseInetXPageProducer (see page 2257)

Procedural types

Creating events

38.5 Constants

The following table lists constants in this documentation.

Constants

Name	Description
sXMLBroker (see page 2301)	This is constant MidProd.sXMLBroker.
sXMLReadyVar (see page 2301)	This is constant MidProd.sXMLReadyVar.

38.5.1 MidProd.sXMLBroker Constant

Pascal

```
sXMLBroker = 'XMLBROKER';
```

C++

```
sXMLBroker = 'XMLBROKER';
```

File

MidProd

Description

This is constant MidProd.sXMLBroker.

38.5.2 MidProd.sXMLReadyVar Constant

Pascal

```
sXMLReadyVar = 'xml_ready';
```

C++

```
sXMLReadyVar = 'xml_ready';
```

File

MidProd

Description

This is constant MidProd.sXMLReadyVar.

39 MPlayer Namespace

This is namespace MPlayer.

Topics

Name	Description
Classes (see page 2305)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 2376)	The following table lists structs, records, enums in this documentation.
Types (see page 2384)	The following table lists types in this documentation.

Classes

	Name	Description
	EMCIDeviceError (see page 2305)	EMCIDeviceError is the exception class for devices that provide a Media Control Interface (MCI) driver.
	TMediaPlayer (see page 2305)	TMediaPlayer controls devices that provide a Media Control Interface (MCI) driver.

Structs, Records, Enums

	Name	Description
	TMPBtnType (see page 2376)	TMPBtnType defines the possible buttons in a TMediaPlayer (see page 2305) component and TButtonSet (see page 2385) represents a set of those buttons.
	TMPButton (see page 2377)	This is record MPlayer.TMPButton.
	TMPDevCaps (see page 2378)	TMPDevCaps defines the capabilities of a TMediaPlayer (see page 2305) component and TMPDevCapsSet (see page 2385) represents a set of those capabilities.
	TMPDeviceTypes (see page 2378)	TMPDeviceTypes identifies a multimedia device.
	TMPGlyph (see page 2380)	This is record MPlayer.TMPGlyph.
	TMPModes (see page 2380)	TMPModes identifies the state of a TMediaPlayer (see page 2305) component.
	TMPNotifyValues (see page 2381)	TMPNotifyValues identifies the types of notifications a media device can give a TMediaPlayer (see page 2305) component.
	TMPTimeFormats (see page 2382)	TMPTimeFormats indicates the format a media player uses to indicate position.

Types

Name	Description
EMPNotify (see page 2384)	EMPNotify is the type of event handlers that respond when the user interacts with a TMediaPlayer (see page 2305) component.
EMPPostNotify (see page 2384)	EMPPostNotify is the type of a media player's OnPostClick event handler.
TButtonSet (see page 2385)	TButtonSet is a set of TMPBtnType (see page 2376) values.

TMediaDevice (see page 2385)	This is type MPlayer.TMediaDevice.
TMPDevCapsSet (see page 2385)	This is type MPlayer.TMPDevCapsSet.

39.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	EMCIDeviceError (see page 2305)	EMCIDeviceError is the exception class for devices that provide a Media Control Interface (MCI) driver.
	TMediaPlayer (see page 2305)	TMediaPlayer controls devices that provide a Media Control Interface (MCI) driver.

39.1.1 EMCIDeviceError Class

EMCIDeviceError is the exception class for devices that provide a Media Control Interface (MCI) driver.

Class Hierarchy

Pascal

```
EMCIDeviceError = class(Exception);
```

C++

```
class EMCIDeviceError : public Exception;
```

File

MPlayer

Description

EMCIDeviceError is raised if an error occurs when accessing a multimedia device. The most common cause for the exception is trying to access a multimedia device before it has been opened with the Open method of TMediaPlayer (see page 2305).

See Also

TMediaPlayer (see page 2305)

39.1.2 TMediaPlayer Class

TMediaPlayer controls devices that provide a Media Control Interface (MCI) driver.

Class Hierarchy

Pascal

```
TMediaPlayer = class(TCustomControl);
```

C++

```
class TMediaPlayer : public TCustomControl;
```

File

MPlayer

Description

The TMediaPlayer component includes a set of buttons (Play (see page 2323), Stop (see page 2330), Eject (see page 2315), and so on) that control a multimedia device such as a CD-ROM drive, MIDI sequencer, or VCR. A multimedia device may be hardware or software.

The media player component contains of multiple buttons. These buttons can be clicked with the mouse, but are not separate objects or button components.

ButtonValueAction

Button	Value	Action
Play (see page 2323)	btPlay	Plays the media player
Pause (see page 2322)	btPause	Pauses playing or recording. If already paused when clicked, resumes playing or recording.
Stop (see page 2330)	btStop	Stops playing or recording
Next (see page 2318)	btNext	Skips to the next track, or to the end if the medium doesn't use tracks
Prev	btPrev	Skips to the previous track, or to the beginning if the medium doesn't use tracks
Step (see page 2330)	btStep	Moves forward a number of frames
Back (see page 2313)	btBack	Moves backward a number of frames
Record	btRecord	Starts recording
Eject (see page 2315)	btEject	Ejects the medium

The multimedia device is played, paused, stopped, and so on when the user clicks the corresponding button on the TMediaPlayer component. The device can also be controlled by the control methods that correspond to the buttons (Play (see page 2323), Pause (see page 2322), Stop (see page 2330), Next (see page 2318), Previous (see page 2327), Step (see page 2330), Back (see page 2313), StartRecording (see page 2329), and Eject (see page 2315)).

The type of multimedia device (such as dtWaveAudio or dtVideodisc) is specified by the DeviceType (see page 2339) property. If the device stores its media in a file, the name of the media file is specified by the FileName (see page 2347) property. If DeviceType (see page 2339) is dtAutoSelect, the media player attempts to determine the type of device from the extension of the file specified by FileName (see page 2347).

To have the media player attempt to open the device specified by DeviceType (see page 2339) automatically when the media player component is created at runtime, set the AutoOpen (see page 2336) property to true.

C++ Examples:

```
//
//This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
//for Open and Stop are present on the form.
//Note: AutoEnable must be set to false to be able to control
//the buttons manually.
//
void __fastcall TForm2::btOpenClick(TObject *Sender)
{
```

```
    TOpenDialog* OpenMediaDialog;

    OpenMediaDialog = new TOpenDialog(this);
    OpenMediaDialog->Filter = "All Video Files (*.avi)|*.avi";
  // There are avi files in
  // C:\Documents and Settings\All Users\Documents\RAD
Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
    if (OpenMediaDialog->Execute())
    {
        /* Assign a file to the media player */
        MediaPlayer1->FileName = OpenMediaDialog->FileName;

        /* Check if the file exists and is not a directory */
        if ((FileExists(OpenMediaDialog->FileName)) &&
           (!DirectoryExists(OpenMediaDialog->FileName)))
        {
            /* Open the files */
            MediaPlayer1->Wait = true;
            MediaPlayer1->Open();
            MediaPlayer1->Play();

            /* Override automatic button controlling */
            MediaPlayer1->EnabledButtons =
                TButtonSet() <<
                TMPBtnType::btPause <<
                TMPBtnType::btStop <<
                TMPBtnType::btPlay;


            /* Enable  the stop button */
            btStop->Enabled = true;
            btOpen->Enabled = false;
        }
    }

    delete OpenMediaDialog;
}

void __fastcall TForm2::btStopClick(TObject *Sender)
{
    /* Stop and close the media */
    MediaPlayer1->Stop();
    MediaPlayer1->Close();

    MediaPlayer1->EnabledButtons = TButtonSet();

    /* Enable open button again */
    btOpen->Enabled = true;
}

void __fastcall TForm2::FormCreate(TObject *Sender)
{
    /* Disable all buttons */
    MediaPlayer1->AutoEnable = false;
    MediaPlayer1->EnabledButtons = TButtonSet();
}

void __fastcall TForm2::MediaPlayer1PostClick(TObject *Sender,
          TMPBtnType Button)
{
    if (Button == TMPBtnType::btStop)
        btStop->Click();
}
```

Delphi Examples:

```
{
```

```
  This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
  for Open and Stop are present on the form.
  Note: AutoEnable must be set to false to be able to control
  the buttons manually.
  }

  procedure TForm2.btOpenClick(Sender: TObject);
  var
    OpenMediaDialog : TOpenDialog;
  begin
    OpenMediaDialog := TOpenDialog.Create(Self);
    OpenMediaDialog.Filter := 'All Video Files (*.avi)|*.avi';
    // There are avi files in
    // C:\Documents and Settings\All Users\Documents\RAD
  Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
    if OpenMediaDialog.Execute() then
    begin
      { Assign a file to the media player }
      MediaPlayer1.FileName := OpenMediaDialog.FileName;

      { Check if the file exists and is not a directory }
      if (FileExists(OpenMediaDialog.FileName)) and
         (not DirectoryExists(OpenMediaDialog.FileName)) then
      begin
        { Open the files }
        MediaPlayer1.Wait := true;
        MediaPlayer1.Open;
        MediaPlayer1.Play;

        { Override automatic button controlling }
        MediaPlayer1.EnabledButtons :=
          [TMPBtnType.btPause, TMPBtnType.btStop, TMPBtnType.btPlay];

        { Enable  the stop button }
        btStop.Enabled := true;
        btOpen.Enabled := false;
      end;
    end;

    OpenMediaDialog.Free;
  end;

  procedure TForm2.btStopClick(Sender: TObject);
  begin
    { Stop and close the media }
    MediaPlayer1.Stop;
    MediaPlayer1.Close;

    MediaPlayer1.EnabledButtons := [];

    { Enable open button again }
    btOpen.Enabled := true;
  end;

  procedure TForm2.FormCreate(Sender: TObject);
  begin
    { Disable all buttons }
    MediaPlayer1.AutoEnable := false;
    MediaPlayer1.EnabledButtons := [];
  end;

  procedure TForm2.MediaPlayer1PostClick(Sender: TObject;
    Button: TMPBtnType);
  begin
    if Button = TMPBtnType.btStop then
       btStop.Click;
```

```
end;
```

See Also

Working with Multimedia

39.1.2.1 TMediaPlayer Members

The following tables list the members exposed by TMediaPlayer.

Protected Methods

	Name	Description
	AutoButtonSet (see page 2312)	Enables or Disables the media player's buttons.
	Click (see page 2313)	Generates an OnClick (see page 2366) event.
	DoNotify (see page 2315)	Generates an OnNotify (see page 2375) event.
V	KeyDown (see page 2317)	Provides responses when the user types arrow keys or the space bar.
V	Loaded (see page 2317)	Initializes the media player after it is loaded from a stream.
	MMNotify (see page 2318)	Responds to a notification that the MCI device has completed an operation.
V	Notification (see page 2319)	Responds when components are about to be inserted or removed.
V	Paint (see page 2322)	Renders the image of the media player.
	PostClick (see page 2326)	Determines the action that takes place when an OnPostClick (see page 2375) event occurs.
V	Updated (see page 2331)	Signals that the media player has finished updating.

Public Methods

	Name	Description
	Back (see page 2313)	Steps backward a number of frames (determined by the value of the Frames (see page 2349) property) in the currently loaded medium.
	Close (see page 2314)	Closes the open multimedia device.
V	Create (see page 2311)	Creates a media player object.
V	Destroy (see page 2311)	Destroys an instance of the media player.
	Eject (see page 2315)	Releases the loaded medium from the open multimedia device.
	Next (see page 2318)	Moves to the beginning of the next track of the currently loaded medium.
	Open (see page 2319)	Opens a multimedia device.
	Pause (see page 2322)	Toggles the open multimedia device on or off.
	PauseOnly (see page 2323)	Pauses the open multimedia device.
	Play (see page 2323)	Plays the media loaded in the open multimedia device.
	Previous (see page 2327)	Sets the current position to the beginning of the previous track if the position was at the beginning of a track when Previous was called.
	Resume (see page 2327)	Resumes playing or recording the currently paused multimedia device.
	Rewind (see page 2328)	Sets the current position to the beginning of the the medium, which is stored in the Start (see page 2360) property.
	Save (see page 2328)	Saves the currently loaded medium to the file specified in the FileName (see page 2347) property.
	StartRecording (see page 2329)	Begins recording from the current Position (see page 2357) or from the position specified in StartPos (see page 2360).
	Step (see page 2330)	Moves forward a number of frames (determined by the Frames (see page 2349) property) in the currently loaded medium.
	Stop (see page 2330)	Halts playing or recording.

Public Properties

	Name	Description
R	Capabilities (see page 2337)	Determines the capabilities of the open multimedia device.
R	DeviceID (see page 2338)	Specifies the device ID for the currently open multimedia device.
	DisplayRect (see page 2340)	Specifies a rectangular area in the control specified by the Display (see page 2339) property that is used to display output from a multimedia device.

	EndPos (see page 2345)	Specifies the position within the currently loaded medium at which to stop playing or recording.
R	Error (see page 2346)	Specifies the MCI error code returned by the media control method.
R	ErrorMessage (see page 2346)	Describes the error code stored in the Error (see page 2346) property.
	Frames (see page 2349)	Specifies the number of frames moved forward or backward.
R	Length (see page 2350)	Specifies the length of the medium in the open multimedia device.
R	Mode (see page 2350)	Indicates the state of the currently open multimedia device.
	Notify (see page 2350)	Determines whether an OnNotify (see page 2375) event is generated.
R	NotifyValue (see page 2351)	Indicates the result of the last media control method that requested a notification.
	Position (see page 2357)	Specifies the current position within the currently loaded medium.
R	Start (see page 2360)	Specifies the starting position within the currently loaded medium.
	StartPos (see page 2360)	Specifies the position within the currently loaded medium from which to begin playing or recording.
	TimeFormat (see page 2362)	Determines the format used to specify position information.
R	TrackLength (see page 2362)	Reports the length of the track.
R	TrackPosition (see page 2362)	Reports the starting position of the track.
R	Tracks (see page 2363)	Specifies the number of playable tracks on the open multimedia device.
	Wait (see page 2365)	Determines whether a media control method returns control to the application only after it has been completed.

Published Events

	Name	Description
	OnClick (see page 2366)	Occurs when the user presses and releases the mouse button while the mouse pointer is over one of the control buttons, or when the user presses Spacebar while the media player control has focus.
	OnContextPopup (see page 2367)	Occurs when the user right-clicks the control or otherwise invokes the popup menu (such as using the keyboard).
	OnEnter (see page 2368)	Occurs when a control receives the input focus.
	OnExit (see page 2370)	Occurs when the input focus shifts away from one control to another.
	OnMouseEnter (see page 2374)	Occurs when the user moves the mouse into a control.
	OnMouseLeave (see page 2374)	Occurs when the user moves the mouse outside of a control.
	OnNotify (see page 2375)	Occurs upon the completion of a media control method.
	OnPostClick (see page 2375)	Occurs after the OnClick (see page 2366) event handler is called.

Published Properties

	Name	Description
	Anchors (see page 2332)	Specifies how the control is anchored to its parent.
	AutoEnable (see page 2333)	Determines whether the media player automatically enables and disables individual buttons in the component.
	AutoOpen (see page 2336)	Determines if the media player is opened automatically when the application is run.
	AutoRewind (see page 2337)	Determines if the media player control rewinds before playing or recording.
	ColoredButtons (see page 2337)	Determines which buttons on the media player control have color.
	Constraints (see page 2338)	Specifies the size constraints for the control.
	DeviceType (see page 2339)	Specifies a multimedia device type to open the media player.
	Display (see page 2339)	Specifies the display window for a multimedia device that uses a window for output.
	DoubleBuffered (see page 2340)	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
	Enabled (see page 2342)	Controls whether the control responds to mouse, keyboard, and timer events.
	EnabledButtons (see page 2343)	Determines which buttons on the media player are usable.
	FileName (see page 2347)	Specifies the opened or saved media file.
	ParentDoubleBuffered (see page 2352)	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
	ParentShowHint (see page 2352)	Determines where a control looks to find out if its Help Hint should be shown.
	PopupMenu (see page 2352)	Identifies the pop-up menu associated with the control.
	Shareable (see page 2358)	Determines whether more than one application can share a multimedia device.

	ShowHint (see page 2358)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	TabOrder (see page 2361)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 2361)	Determines if the user can tab to a control.
	Visible (see page 2363)	Determines whether the component appears onscreen.
	VisibleButtons (see page 2365)	Indicates which of the buttons on the media player are visible.

39.1.2.2 TMediaPlayer.Create Constructor

Creates a media player object.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TMediaPlayer(TComponent * AOwner);
```

Description

Call Create to instantiate a media player at runtime. For media players created at design time, Create is called automatically.

Create allocates memory for a media player and calls the inherited Create. Then it loads the bitmaps for the media player's buttons and initializes its properties, setting AutoEnable (see page 2333), AutoRewind (see page 2337), Colored, Enabled (see page 2342), and Visible (see page 2363) to true, AutoOpen (see page 2336) to false, and DeviceType (see page 2339) to dtAutoSelect.

See Also

Create

ParentWindow

39.1.2.3 TMediaPlayer.Destroy Destructor

Destroys an instance of the media player.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TMediaPlayer();
```

Description

Do not call Destroy directly in an application. Call Free instead. Free checks that the TMediaPlayer (see page 2305) reference is not nil before it calls Destroy.

Destroy first verifies that no device is open, then destroys the bitmaps used for the media player's buttons and calls the inherited Destroy.

See Also

Destroy

System::TObject::Free

39.1.2.4 TMediaPlayer Methods

The methods of the TMediaPlayer class are listed here.

Protected Methods

	Name	Description
	AutoButtonSet (see page 2312)	Enables or Disables the media player's buttons.
	Click (see page 2313)	Generates an OnClick (see page 2366) event.
	DoNotify (see page 2315)	Generates an OnNotify (see page 2375) event.
	KeyDown (see page 2317)	Provides responses when the user types arrow keys or the space bar.
	Loaded (see page 2317)	Initializes the media player after it is loaded from a stream.
	MMNotify (see page 2318)	Responds to a notification that the MCI device has completed an operation.
	Notification (see page 2319)	Responds when components are about to be inserted or removed.
	Paint (see page 2322)	Renders the image of the media player.
	PostClick (see page 2326)	Determines the action that takes place when an OnPostClick (see page 2375) event occurs.
	Updated (see page 2331)	Signals that the media player has finished updating.

Public Methods

	Name	Description
	Back (see page 2313)	Steps backward a number of frames (determined by the value of the Frames (see page 2349) property) in the currently loaded medium.
	Close (see page 2314)	Closes the open multimedia device.
	Eject (see page 2315)	Releases the loaded medium from the open multimedia device.
	Next (see page 2318)	Moves to the beginning of the next track of the currently loaded medium.
	Open (see page 2319)	Opens a multimedia device.
	Pause (see page 2322)	Toggles the open multimedia device on or off.
	PauseOnly (see page 2323)	Pauses the open multimedia device.
	Play (see page 2323)	Plays the media loaded in the open multimedia device.
	Previous (see page 2327)	Sets the current position to the beginning of the previous track if the position was at the beginning of a track when Previous was called.
	Resume (see page 2327)	Resumes playing or recording the currently paused multimedia device.
	Rewind (see page 2328)	Sets the current position to the beginning of the the medium, which is stored in the Start (see page 2360) property.
	Save (see page 2328)	Saves the currently loaded medium to the file specified in the FileName (see page 2347) property.
	StartRecording (see page 2329)	Begins recording from the current Position (see page 2357) or from the position specified in StartPos (see page 2360).
	Step (see page 2330)	Moves forward a number of frames (determined by the Frames (see page 2349) property) in the currently loaded medium.
	Stop (see page 2330)	Halts playing or recording.

39.1.2.4.1 TMediaPlayer.AutoButtonSet Method

Enables or Disables the media player's buttons.

Pascal

```
procedure AutoButtonSet(Btn: TMPBtnType); dynamic;
```

C++

```
__fastcall AutoButtonSet(TMPBtnType Btn);
```

Description

AutoButtonSet implements the automatic enabling and disabling of the media player's buttons when the AutoEnable (see page 2333) property is true. The Btn parameter indicates which of the media player's buttons the user pressed.

See Also

AutoEnable (see page 2333)

EnabledButtons (see page 2343)

39.1.2.4.2 TMediaPlayer.Back Method

Steps backward a number of frames (determined by the value of the Frames (see page 2349) property) in the currently loaded medium.

Pascal

```
procedure Back;
```

C++

```
__fastcall Back();
```

Description

Back is called when the Back button on the media player control is clicked at runtime.

Upon completion, Back stores a numerical error code in the Error (see page 2346) property and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Back method has been completed. The Notify (see page 2350) property determines whether Back generates an OnNotify (see page 2375) event.

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Wait (see page 2365)

Adding Audio and/or Video Clips to an Application

39.1.2.4.3 TMediaPlayer.Click Method

Generates an OnClick (see page 2366) event.

Pascal

```
procedure Click(Button: TMPBtnType; var DoDefault: Boolean); dynamic; reintroduce;
```

C++

```
__fastcall Click(TMPBtnType Button, Boolean DoDefault);
```

Description

Click for media-player components is the implementation method for the OnClick (see page 2366) event. By default, Click does nothing other than call any event handler attached to the media player's OnClick (see page 2366) event. Click can be overridden in descendant classes to customize responses to clicks.

See Also

OnClick (see page 2366)

39.1.2.4.4 TMediaPlayer.Close Method

Closes the open multimedia device.

Pascal

```
procedure Close;
```

C++

```
__fastcall Close();
```

Description

Upon completion, Close stores a numerical error code in the Error (see page 2346) property, and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Close method is completed. The Notify (see page 2350) property determines whether Close generates an OnNotify (see page 2375) event.

Close is called automatically when the application is terminated.

C++ Examples:

```
/*
This code ejects the CD from the CD-ROM player and shuts
down the media player after 10 seconds. For the code to run
correctly, you must have your CD audio device installed
correctly, and the device must have software-ejecting
capabilities.
*/

Word TimeOver;

void __fastcall TForm1::FormClick(TObject *Sender)
{
  MediaPlayer1->DeviceType = dtCDAudio;
  MediaPlayer1->Open();
  MediaPlayer1->Play();
  Timer1->Enabled = true;
  TimeOver = 0;
}

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
  if (TimeOver == 10)
  {
    MediaPlayer1->Eject();
    MediaPlayer1->Close();
    // disable the timer, as we are done
    Timer1->Enabled = false;
  }
  else
    TimeOver++;
}
```

Delphi Examples:

```
{
This code ejects the CD from the CD-ROM player and shuts
down the media player after 10 seconds. For the code to run
correctly, you must have your CD audio device installed
correctly, and the device must have software-ejecting
capabilities.
}
var
```

```
    TimerOver: Word;

  procedure TForm1.FormClick(Sender: TObject);
  begin
    MediaPlayer1.DeviceType := dtCDAudio;
    MediaPlayer1.Open;
    MediaPlayer1.Play;
  end;

  procedure TForm1.Timer1Timer(Sender: TObject);
  begin
    if TimeOver = 10 then
    begin
      MediaPlayer1.Eject;
      MediaPlayer1.Close;
      { disable the timer, as we are done }
      Timer1.Enabled := False;
    end
    else
      if (MediaPlayer1.Mode = mpOpen) then
       Inc(TimeOver);
  end;
```

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Wait (see page 2365)

39.1.2.4.5 TMediaPlayer.DoNotify Method

Generates an OnNotify (see page 2375) event.

Pascal

```
procedure DoNotify; dynamic;
```

C++

```
__fastcall DoNotify();
```

Description

DoNotify is the implementation method for a media-player component's OnNotify (see page 2375) event. The media player's MMNotify (see page 2318) method calls DoNotify after correcting the enabling and disabling of buttons and setting internal flags from the values passed in the notification message parameters.

By default, DoNotify does nothing except call any event handler attached to the media player's OnNotify (see page 2375) event. DoNotify can be overridden to provide other responses in addition to the inherited event-handler call.

See Also

MMNotify (see page 2318)

OnNotify (see page 2375)

39.1.2.4.6 TMediaPlayer.Eject Method

Releases the loaded medium from the open multimedia device.

Pascal

```
procedure Eject;
```

C++

```
__fastcall Eject();
```

Description

Eject is called when the Eject button on the media player control is clicked at runtime. It ejects the loaded medium from the open multimedia device.

Upon completion, Eject stores a numerical error code in the Error (see page 2346) property, and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Eject method has been completed. The Notify (see page 2350) property determines whether Eject generates an OnNotify (see page 2375) event.

C++ Examples:

```
/*
This code ejects the CD from the CD-ROM player and shuts
down the media player after 10 seconds. For the code to run
correctly, you must have your CD audio device installed
correctly, and the device must have software-ejecting
capabilities.
*/

Word TimeOver;

void __fastcall TForm1::FormClick(TObject *Sender)
{
  MediaPlayer1->DeviceType = dtCDAudio;
  MediaPlayer1->Open();
  MediaPlayer1->Play();
  Timer1->Enabled = true;
  TimeOver = 0;
}

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
  if (TimeOver == 10)
  {
    MediaPlayer1->Eject();
    MediaPlayer1->Close();
    // disable the timer, as we are done
    Timer1->Enabled = false;
  }
  else
    TimeOver++;
}
```

Delphi Examples:

```
{
This code ejects the CD from the CD-ROM player and shuts
down the media player after 10 seconds. For the code to run
correctly, you must have your CD audio device installed
correctly, and the device must have software-ejecting
capabilities.
}
var
  TimerOver: Word;

procedure TForm1.FormClick(Sender: TObject);
begin
```

```
    MediaPlayer1.DeviceType := dtCDAudio;
    MediaPlayer1.Open;
    MediaPlayer1.Play;
  end;

  procedure TForm1.Timer1Timer(Sender: TObject);
  begin
    if TimeOver = 10 then
    begin
      MediaPlayer1.Eject;
      MediaPlayer1.Close;
      { disable the timer, as we are done }
      Timer1.Enabled := False;
    end
    else
      if (MediaPlayer1.Mode = mpOpen) then
       Inc(TimeOver);
  end;
```

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Wait (see page 2365)

Adding Audio and/or Video Clips to an Application

39.1.2.4.7 TMediaPlayer.KeyDown Method

Provides responses when the user types arrow keys or the space bar.

Pascal

```
procedure KeyDown(var Key: Word; Shift: TShiftState); override;
```

C++

```
virtual __fastcall KeyDown(Word Key, TShiftState Shift);
```

Description

KeyDown overrides the inherited method to implement the media player's response to certain keystrokes. It changes the current button when the user types the left or right arrow keys. When the user presses the space bar, KeyDown simulates a click on the current button.

See Also

EnabledButtons (see page 2343)

KeyDown

39.1.2.4.8 TMediaPlayer.Loaded Method

Initializes the media player after it is loaded from a stream.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

Loaded provides an opportunity for a component to initialize itself after all its parts have loaded from a stream. TMediaPlayer (see page 2305) overrides the inherited method to implement the AutoOpen (see page 2336) property.

See Also

Create (see page 2311)

AutoOpen (see page 2336)

39.1.2.4.9 TMediaPlayer.MMNotify Method

Responds to a notification that the MCI device has completed an operation.

Pascal

```
procedure MMNotify(var Message: TMessage); message MM_MCINOTIFY;
```

C++

```
__fastcall MMNotify(TMessage Message);
```

Description

MMNotify responds to a notification message from Windows that a multimedia command completed by updating button states if the AutoEnable (see page 2333) property is true, setting internal flags to reflect the outcome of the command, then calling the DoNotify (see page 2315) method to trigger the OnNotify (see page 2375) event and any other special responses.

See Also

AutoEnable (see page 2333)

DoNotify (see page 2315)

OnNotify (see page 2375)

39.1.2.4.10 TMediaPlayer.Next Method

Moves to the beginning of the next track of the currently loaded medium.

Pascal

```
procedure Next;
```

C++

```
__fastcall Next();
```

Description

If the current position is at the last track when Next is called, Next makes the current position the beginning of the last track. If the multimedia device doesn't use tracks, Next goes to the end of the medium. Next is called when the Next button on the media player control is clicked at runtime.

Upon completion, Next stores a numerical error code in the Error (see page 2346) property, and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Next method has completed. The Notify (see page 2350) property determines whether Next generates an OnNotify (see page 2375) event.

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Wait (see page 2365)

Adding Audio and/or Video Clips to an Application

39.1.2.4.11 TMediaPlayer.Notification Method

Responds when components are about to be inserted or removed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

By default, components pass along the notification to their owned components, if any. Notification calls the inherited Notification and then, if the Display (see page 2339) window is deleted at runtime, it sets the video display back to original window.

See Also

Notification

Display (see page 2339)

39.1.2.4.12 TMediaPlayer.Open Method

Opens a multimedia device.

Pascal

```
procedure Open;
```

C++

```
__fastcall Open();
```

Description

Use Open to open a multimedia device. The multimedia device type must be specified in the DeviceType (see page 2339) property before a device can be opened.

Upon completion, Open stores a numerical error code in the Error (see page 2346) property, and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Open method is completed. The Notify (see page 2350) property determines whether Open generates an OnNotify (see page 2375) event.

C++ Examples:

```
//
//This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
//for Open and Stop are present on the form.
//Note: AutoEnable must be set to false to be able to control
```

```
//the buttons manually.
//
void __fastcall TForm2::btOpenClick(TObject *Sender)
{
    TOpenDialog* OpenMediaDialog;

    OpenMediaDialog = new TOpenDialog(this);
    OpenMediaDialog->Filter = "All Video Files (*.avi)|*.avi";
  // There are avi files in
  // C:\Documents and Settings\All Users\Documents\RAD
Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
    if (OpenMediaDialog->Execute())
    {
        /* Assign a file to the media player */
        MediaPlayer1->FileName = OpenMediaDialog->FileName;

        /* Check if the file exists and is not a directory */
        if ((FileExists(OpenMediaDialog->FileName)) &&
           (!DirectoryExists(OpenMediaDialog->FileName)))
        {
            /* Open the files */
            MediaPlayer1->Wait = true;
            MediaPlayer1->Open();
            MediaPlayer1->Play();

            /* Override automatic button controlling */
            MediaPlayer1->EnabledButtons =
                TButtonSet() <<
                TMPBtnType::btPause <<
                TMPBtnType::btStop <<
                TMPBtnType::btPlay;


            /* Enable  the stop button */
            btStop->Enabled = true;
            btOpen->Enabled = false;
        }
    }

    delete OpenMediaDialog;
}

void __fastcall TForm2::btStopClick(TObject *Sender)
{
    /* Stop and close the media */
    MediaPlayer1->Stop();
    MediaPlayer1->Close();

    MediaPlayer1->EnabledButtons = TButtonSet();

    /* Enable open button again */
    btOpen->Enabled = true;
}

void __fastcall TForm2::FormCreate(TObject *Sender)
{
    /* Disable all buttons */
    MediaPlayer1->AutoEnable = false;
    MediaPlayer1->EnabledButtons = TButtonSet();
}

void __fastcall TForm2::MediaPlayer1PostClick(TObject *Sender,
          TMPBtnType Button)
{
    if (Button == TMPBtnType::btStop)
        btStop->Click();
```

```
}
```

Delphi Examples:

```
{
This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
for Open and Stop are present on the form.
Note: AutoEnable must be set to false to be able to control
the buttons manually.
}

procedure TForm2.btOpenClick(Sender: TObject);
var
  OpenMediaDialog : TOpenDialog;
begin
  OpenMediaDialog := TOpenDialog.Create(Self);
  OpenMediaDialog.Filter := 'All Video Files (*.avi)|*.avi';
  // There are avi files in
  // C:\Documents and Settings\All Users\Documents\RAD
Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
  if OpenMediaDialog.Execute() then
  begin
    { Assign a file to the media player }
    MediaPlayer1.FileName := OpenMediaDialog.FileName;

    { Check if the file exists and is not a directory }
    if (FileExists(OpenMediaDialog.FileName)) and
       (not DirectoryExists(OpenMediaDialog.FileName)) then
    begin
      { Open the files }
      MediaPlayer1.Wait := true;
      MediaPlayer1.Open;
      MediaPlayer1.Play;

      { Override automatic button controlling }
      MediaPlayer1.EnabledButtons :=
        [TMPBtnType.btPause, TMPBtnType.btStop, TMPBtnType.btPlay];

      { Enable  the stop button }
      btStop.Enabled := true;
      btOpen.Enabled := false;
    end;
  end;

  OpenMediaDialog.Free;
end;

procedure TForm2.btStopClick(Sender: TObject);
begin
  { Stop and close the media }
  MediaPlayer1.Stop;
  MediaPlayer1.Close;

  MediaPlayer1.EnabledButtons := [];

  { Enable open button again }
  btOpen.Enabled := true;
end;

procedure TForm2.FormCreate(Sender: TObject);
begin
  { Disable all buttons }
  MediaPlayer1.AutoEnable := false;
  MediaPlayer1.EnabledButtons := [];
end;

procedure TForm2.MediaPlayer1PostClick(Sender: TObject;
```

```
  Button: TMPBtnType);
begin
  if Button = TMPBtnType.btStop then
     btStop.Click;
end;
```

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Wait (see page 2365)

Adding Audio and/or Video Clips to an Application

39.1.2.4.13 TMediaPlayer.Paint Method

Renders the image of the media player.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

When the media player component receives a WM_PAINT (see page 1750) message, like other custom controls, it responds by initializing the control's canvas and calling Paint. Paint draws the frame and all visible buttons.

When creating a descendant media player class, Paint can be overridden to draw the image of the corresponding media player object.

See Also

VisibleButtons (see page 2365)

39.1.2.4.14 TMediaPlayer.Pause Method

Toggles the open multimedia device on or off.

Pascal

```
procedure Pause;
```

C++

```
__fastcall Pause();
```

Description

If the device is already paused when Pause is called, the device resumes playing or recording by calling the Resume (see page 2327) method. Pause is called when the Pause button on the media player control is clicked at runtime.

Upon completion, Pause stores a numerical error code in the Error (see page 2346) property and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Pause method has completed. The Notify (see page 2350) property determines whether Pause generates an OnNotify (see page 2375) event.

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Resume (see page 2327)

Wait (see page 2365)

Adding Audio and/or Video Clips to an Application

39.1.2.4.15 TMediaPlayer.PauseOnly Method

Pauses the open multimedia device.

Pascal

```
procedure PauseOnly;
```

C++

```
__fastcall PauseOnly();
```

Description

If the device is already paused when PauseOnly is called, the device will remain paused.

Upon completion, PauseOnly stores a numerical error code in the Error (see page 2346) property and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the PauseOnly method has completed. The Notify (see page 2350) property determines whether PauseOnly generates an OnNotify (see page 2375) event.

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Wait (see page 2365)

39.1.2.4.16 TMediaPlayer.Play Method

Plays the media loaded in the open multimedia device.

Pascal

```
procedure Play;
```

C++

```
__fastcall Play();
```

Description

Play is called when the Play button on the media player control is clicked at runtime.

Upon completion, Play stores a numerical error code in the Error (see page 2346) property and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Play method has completed. The Notify (see page 2350) property determines whether Play generates an OnNotify (see page 2375) event.

If the StartPos (see page 2360) property is set, playing starts at the position specified in StartPos (see page 2360). Otherwise, playing starts at the current position, specified in the Position (see page 2357) property. Similarly, if the EndPos (see page 2345) property is set, playing stops at the position specified in EndPos (see page 2345). Otherwise, playing stops at the end of the medium.

Whether the medium (specified in the Position (see page 2357) property) is rewound before playing starts depends on the AutoRewind (see page 2337) property.

C++ Examples:

```
//
//This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
//for Open and Stop are present on the form.
//Note: AutoEnable must be set to false to be able to control
//the buttons manually.
//
void __fastcall TForm2::btOpenClick(TObject *Sender)
{
    TOpenDialog* OpenMediaDialog;

    OpenMediaDialog = new TOpenDialog(this);
    OpenMediaDialog->Filter = "All Video Files (*.avi)|*.avi";
  // There are avi files in
  // C:\Documents and Settings\All Users\Documents\RAD
Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
    if (OpenMediaDialog->Execute())
    {
        /* Assign a file to the media player */
        MediaPlayer1->FileName = OpenMediaDialog->FileName;

        /* Check if the file exists and is not a directory */
        if ((FileExists(OpenMediaDialog->FileName)) &&
           (!DirectoryExists(OpenMediaDialog->FileName)))
        {
            /* Open the files */
            MediaPlayer1->Wait = true;
            MediaPlayer1->Open();
            MediaPlayer1->Play();

            /* Override automatic button controlling */
            MediaPlayer1->EnabledButtons =
                TButtonSet() <<
                TMPBtnType::btPause <<
                TMPBtnType::btStop <<
                TMPBtnType::btPlay;


            /* Enable  the stop button */
            btStop->Enabled = true;
            btOpen->Enabled = false;
        }
    }

    delete OpenMediaDialog;
}

void __fastcall TForm2::btStopClick(TObject *Sender)
{
    /* Stop and close the media */
```

```
    MediaPlayer1->Stop();
    MediaPlayer1->Close();

    MediaPlayer1->EnabledButtons = TButtonSet();

    /* Enable open button again */
    btOpen->Enabled = true;
}

void __fastcall TForm2::FormCreate(TObject *Sender)
{
    /* Disable all buttons */
    MediaPlayer1->AutoEnable = false;
    MediaPlayer1->EnabledButtons = TButtonSet();
}

void __fastcall TForm2::MediaPlayer1PostClick(TObject *Sender,
          TMPBtnType Button)
{
    if (Button == TMPBtnType::btStop)
        btStop->Click();
}
```

Delphi Examples:

```
{
This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
for Open and Stop are present on the form.
Note: AutoEnable must be set to false to be able to control
the buttons manually.
}

procedure TForm2.btOpenClick(Sender: TObject);
var
  OpenMediaDialog : TOpenDialog;
begin
  OpenMediaDialog := TOpenDialog.Create(Self);
  OpenMediaDialog.Filter := 'All Video Files (*.avi)|*.avi';
  // There are avi files in
  // C:\Documents and Settings\All Users\Documents\RAD
Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
  if OpenMediaDialog.Execute() then
  begin
    { Assign a file to the media player }
    MediaPlayer1.FileName := OpenMediaDialog.FileName;

    { Check if the file exists and is not a directory }
    if (FileExists(OpenMediaDialog.FileName)) and
       (not DirectoryExists(OpenMediaDialog.FileName)) then
    begin
      { Open the files }
      MediaPlayer1.Wait := true;
      MediaPlayer1.Open;
      MediaPlayer1.Play;

      { Override automatic button controlling }
      MediaPlayer1.EnabledButtons :=
        [TMPBtnType.btPause, TMPBtnType.btStop, TMPBtnType.btPlay];

      { Enable  the stop button }
      btStop.Enabled := true;
      btOpen.Enabled := false;
    end;
  end;

  OpenMediaDialog.Free;
end;
```

```
procedure TForm2.btStopClick(Sender: TObject);
begin
  { Stop and close the media }
  MediaPlayer1.Stop;
  MediaPlayer1.Close;

  MediaPlayer1.EnabledButtons := [];

  { Enable open button again }
  btOpen.Enabled := true;
end;

procedure TForm2.FormCreate(Sender: TObject);
begin
  { Disable all buttons }
  MediaPlayer1.AutoEnable := false;
  MediaPlayer1.EnabledButtons := [];
end;

procedure TForm2.MediaPlayer1PostClick(Sender: TObject;
  Button: TMPBtnType);
begin
  if Button = TMPBtnType.btStop then
     btStop.Click;
end;
```

See Also

AutoRewind (see page 2337)

EndPos (see page 2345)

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Position (see page 2357)

Resume (see page 2327)

StartPos (see page 2360)

Wait (see page 2365)

Adding Audio and/or Video Clips to an Application

39.1.2.4.17 TMediaPlayer.PostClick Method

Determines the action that takes place when an OnPostClick (see page 2375) event occurs.

Pascal

```
procedure PostClick(Button: TMPBtnType); dynamic;
```

C++

```
__fastcall PostClick(TMPBtnType Button);
```

Description

PostClick is the implementation method for the OnPostClick (see page 2375) event. By default, PostClick does nothing other than call any event handler attached to the media player's OnPostClick (see page 2375) event. PostClick can be overridden in

descendant types to customize responses to clicks.

See Also

OnPostClick (see page 2375)

39.1.2.4.18 TMediaPlayer.Previous Method

Sets the current position to the beginning of the previous track if the position was at the beginning of a track when Previous was called.

Pascal

```
procedure Previous;
```

C++

```
__fastcall Previous();
```

Description

If the position is at the first track or somewhere other than the beginning of a track when Previous was called, Previous sets the current position to the beginning of the current track. If the device doesn't use tracks, Previous sets the current position to the beginning of the medium, which is specified in the Start (see page 2360) property. Previous is called when the Previous button on the media player control is clicked at runtime.

Upon completion, Previous stores a numerical error code in the Error (see page 2346) property and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Previous method has completed. The Notify (see page 2350) property determines whether Previous generates an OnNotify (see page 2375) event.

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Start (see page 2360)

Wait (see page 2365)

Adding Audio and/or Video Clips to an Application

39.1.2.4.19 TMediaPlayer.Resume Method

Resumes playing or recording the currently paused multimedia device.

Pascal

```
procedure Resume;
```

C++

```
__fastcall Resume();
```

Description

Resume is called when the Pause (see page 2322) button on the media player control is clicked at runtime, when the device is paused.

Upon completion, Resume stores a numerical error code in the Error (see page 2346) property, and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Resume method has completed. The Notify (see page 2350) property determines whether Resume generates an OnNotify (see page 2375) event.

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Wait (see page 2365)

39.1.2.4.20 TMediaPlayer.Rewind Method

Sets the current position to the beginning of the the medium, which is stored in the Start (see page 2360) property.

Pascal

```
procedure Rewind;
```

C++

```
__fastcall Rewind();
```

Description

Upon completion, Rewind stores a numerical error code in the Error (see page 2346) property, and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Rewind method has completed. The Notify (see page 2350) property determines whether Rewind generates an OnNotify (see page 2375) event.

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Start (see page 2360)

Wait (see page 2365)

39.1.2.4.21 TMediaPlayer.Save Method

Saves the currently loaded medium to the file specified in the FileName (see page 2347) property.

Pascal

```
procedure Save;
```

C++

```
__fastcall Save();
```

Description

Save is ignored for devices that don't use media stored in files (videodiscs, for example).

Upon completion, Save stores a numerical error code in the Error (see page 2346) property, and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Save method has completed. The Notify (see page 2350) property determines whether Save generates an OnNotify (see page 2375) event.

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

FileName (see page 2347)

Notify (see page 2350)

OnNotify (see page 2375)

Wait (see page 2365)

39.1.2.4.22 TMediaPlayer.StartRecording Method

Begins recording from the current Position (see page 2357) or from the position specified in StartPos (see page 2360).

Pascal

```
procedure StartRecording;
```

C++

```
__fastcall StartRecording();
```

Description

StartRecording is called when the Record button on the media player control is clicked at runtime.

Upon completion, StartRecording stores a numerical error code in the Error (see page 2346) property and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the StartRecording method has completed. The Notify (see page 2350) property determines whether StartRecording generates an OnNotify (see page 2375) event.

By default, the Notify (see page 2350) property becomes true, and the Wait (see page 2365) property becomes false upon completion of the StartRecording method. However, if these properties have been set to specific values prior to calling StartRecording, they remain unchanged.

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Notify (see page 2350)

OnNotify (see page 2375)

Position (see page 2357)

StartPos (see page 2360)

Wait (see page 2365)

39.1.2.4.23 TMediaPlayer.Step Method

Moves forward a number of frames (determined by the Frames (see page 2349) property) in the currently loaded medium.

Pascal

```
procedure Step;
```

C++

```
__fastcall Step();
```

Description

Step is called when the Step button on the media player control is clicked at runtime.

Upon completion, Step stores a numerical error code in the Error (see page 2346) property and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Step method has completed. The Notify (see page 2350) property determines whether Step generates an OnNotify (see page 2375) event.

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

Frames (see page 2349)

Notify (see page 2350)

OnNotify (see page 2375)

Wait (see page 2365)

Adding Audio and/or Video Clips to an Application

39.1.2.4.24 TMediaPlayer.Stop Method

Halts playing or recording.

Pascal

```
procedure Stop;
```

C++

```
__fastcall Stop();
```

Description

Stop is called when the Stop button on the media player control is clicked at runtime.

Upon completion, Stop stores a numerical error code in the Error (see page 2346) property, and the corresponding error message in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before the Stop method has completed. The Notify (see page 2350) property determines whether Stop generates an OnNotify (see page 2375) event.

See Also

Error (see page 2346)

ErrorMessage (see page 2346)

FileName (see page 2347)

Notify (see page 2350)

OnNotify (see page 2375)

Wait (see page 2365)

Adding Audio and/or Video Clips to an Application

39.1.2.4.25 TMediaPlayer.Updated Method

Signals that the media player has finished updating.

Pascal

```
procedure Updated; override;
```

C++

```
virtual __fastcall Updated();
```

Description

Updated calls the inherited Updated before it modifies the layout of the media player buttons and forces a repaint.

See Also

Updated

39.1.2.5 TMediaPlayer Properties

The properties of the TMediaPlayer class are listed here.

Public Properties

	Name	Description
R	Capabilities (see page 2337)	Determines the capabilities of the open multimedia device.
R	DeviceID (see page 2338)	Specifies the device ID for the currently open multimedia device.
	DisplayRect (see page 2340)	Specifies a rectangular area in the control specified by the Display (see page 2339) property that is used to display output from a multimedia device.
	EndPos (see page 2345)	Specifies the position within the currently loaded medium at which to stop playing or recording.
R	Error (see page 2346)	Specifies the MCI error code returned by the media control method.
R	ErrorMessage (see page 2346)	Describes the error code stored in the Error (see page 2346) property.
	Frames (see page 2349)	Specifies the number of frames moved forward or backward.
R	Length (see page 2350)	Specifies the length of the medium in the open multimedia device.
R	Mode (see page 2350)	Indicates the state of the currently open multimedia device.
	Notify (see page 2350)	Determines whether an OnNotify (see page 2375) event is generated.
R	NotifyValue (see page 2351)	Indicates the result of the last media control method that requested a notification.
	Position (see page 2357)	Specifies the current position within the currently loaded medium.
R	Start (see page 2360)	Specifies the starting position within the currently loaded medium.
	StartPos (see page 2360)	Specifies the position within the currently loaded medium from which to begin playing or recording.
	TimeFormat (see page 2362)	Determines the format used to specify position information.
R	TrackLength (see page 2362)	Reports the length of the track.
R	TrackPosition (see page 2362)	Reports the starting position of the track.
R	Tracks (see page 2363)	Specifies the number of playable tracks on the open multimedia device.
	Wait (see page 2365)	Determines whether a media control method returns control to the application only after it has been completed.

Published Properties

	Name	Description
	Anchors (see page 2332)	Specifies how the control is anchored to its parent.
	AutoEnable (see page 2333)	Determines whether the media player automatically enables and disables individual buttons in the component.
	AutoOpen (see page 2336)	Determines if the media player is opened automatically when the application is run.
	AutoRewind (see page 2337)	Determines if the media player control rewinds before playing or recording.
	ColoredButtons (see page 2337)	Determines which buttons on the media player control have color.
	Constraints (see page 2338)	Specifies the size constraints for the control.
	DeviceType (see page 2339)	Specifies a multimedia device type to open the media player.
	Display (see page 2339)	Specifies the display window for a multimedia device that uses a window for output.
	DoubleBuffered (see page 2340)	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
	Enabled (see page 2342)	Controls whether the control responds to mouse, keyboard, and timer events.
	EnabledButtons (see page 2343)	Determines which buttons on the media player are usable.
	FileName (see page 2347)	Specifies the opened or saved media file.
	ParentDoubleBuffered (see page 2352)	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
	ParentShowHint (see page 2352)	Determines where a control looks to find out if its Help Hint should be shown.
	PopupMenu (see page 2352)	Identifies the pop-up menu associated with the control.
	Shareable (see page 2358)	Determines whether more than one application can share a multimedia device.
	ShowHint (see page 2358)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	TabOrder (see page 2361)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 2361)	Determines if the user can tab to a control.
	Visible (see page 2363)	Determines whether the component appears onscreen.
	VisibleButtons (see page 2365)	Indicates which of the buttons on the media player are visible.

39.1.2.5.1 TMediaPlayer.Anchors Property

Specifies how the control is anchored to its parent.

Pascal

```
property Anchors;
```

C++

```
__property Anchors;
```

Description

Use Anchors to ensure that a control maintains its current position relative to an edge of its parent, even if the parent is resized. When its parent is resized, the control holds its position relative to the edges to which it is anchored.

If a control is anchored to opposite edges of its parent, the control stretches when its parent is resized. For example, if a control has its Anchors property set to [akLeft, akRight], the control stretches when the width of its parent changes.

Anchors is enforced only when the parent is resized. Thus, for example, if a control is anchored to opposite edges of a form at design time and the form is created in a maximized state, the control is not stretched because the form is not resized after the control is created.

Note: If a control should maintain contact with three edges of its parent (hugging one side of the parent and stretching the length of that side), use the Align property instead. Unlike Anchors, Align allows controls to adjust to changes in the size of other aligned sibling controls as well as changes to the parent's size.

See Also

Parent

Align

39.1.2.5.2 TMediaPlayer.AutoEnable Property

Determines whether the media player automatically enables and disables individual buttons in the component.

Pascal

```
property AutoEnable: Boolean;
```

C++

```
__property Boolean AutoEnable;
```

Description

Use AutoEnable to automatically enable or disable the control buttons on the media player. If AutoEnable is true, the media player automatically enables or disables its control buttons. The media player determines which buttons to enable or disable by the current mode specified in the Mode (see page 2350) property, and the current multimedia device type specified in the DeviceType (see page 2339) property.

AutoEnable overrides the EnabledButtons (see page 2343) property. The buttons enabled or disabled automatically by the media player supersede any buttons enabled or disabled with EnabledButtons (see page 2343).

If AutoEnable is false, the media player does not enable or disable buttons. The buttons must enabled or disabled with the EnabledButtons (see page 2343) property.

The following table shows whether buttons are automatically enabled or disabled for each device mode:

Button	Play (see page 2323)	Record	Pause (see page 2322)	Stop (see page 2330)	Not Open (see page 2319)
Back (see page 2313)	Enabled (see page 2342)	Enabled (see page 2342)	Enabled (see page 2342)	Enabled (see page 2342)	Disabled
Eject (see page 2315)	Enabled (see page 2342)	Enabled (see page 2342)	Enabled (see page 2342)	Enabled (see page 2342)	Disabled
Next (see page 2318)	Enabled (see page 2342)	Enabled (see page 2342)	Enabled (see page 2342)	Enabled (see page 2342)	Disabled
Pause (see page 2322)	Enabled (see page 2342)	Enabled (see page 2342)	Enabled (see page 2342)	Disabled	Disabled
Play (see page 2323)	Disabled	Disabled	Enabled (see page 2342)	Enabled (see page 2342)	Disabled
Prev	Enabled (see page 2342)	Enabled (see page 2342)	Enabled (see page 2342)	Enabled (see page 2342)	Disabled
Record	Disabled	Disabled	Enabled (see page 2342)	Enabled (see page 2342)	Disabled
Step (see page 2330)	Enabled (see page 2342)	Enabled (see page 2342)	Enabled (see page 2342)	Enabled (see page 2342)	Disabled
Stop (see page 2330)	Enabled (see page 2342)	Enabled (see page 2342)	Disabled	Disabled	Disabled

C++ Examples:

```
//
```

```
//This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
//for Open and Stop are present on the form.
//Note: AutoEnable must be set to false to be able to control
//the buttons manually.
//
void __fastcall TForm2::btOpenClick(TObject *Sender)
{
    TOpenDialog* OpenMediaDialog;

    OpenMediaDialog = new TOpenDialog(this);
    OpenMediaDialog->Filter = "All Video Files (*.avi)|*.avi";
  // There are avi files in
  // C:\Documents and Settings\All Users\Documents\RAD
Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
    if (OpenMediaDialog->Execute())
    {
        /* Assign a file to the media player */
        MediaPlayer1->FileName = OpenMediaDialog->FileName;

        /* Check if the file exists and is not a directory */
        if ((FileExists(OpenMediaDialog->FileName)) &&
           (!DirectoryExists(OpenMediaDialog->FileName)))
        {
            /* Open the files */
            MediaPlayer1->Wait = true;
            MediaPlayer1->Open();
            MediaPlayer1->Play();

            /* Override automatic button controlling */
            MediaPlayer1->EnabledButtons =
                TButtonSet() <<
                TMPBtnType::btPause <<
                TMPBtnType::btStop <<
                TMPBtnType::btPlay;


            /* Enable  the stop button */
            btStop->Enabled = true;
            btOpen->Enabled = false;
        }
    }

    delete OpenMediaDialog;
}

void __fastcall TForm2::btStopClick(TObject *Sender)
{
    /* Stop and close the media */
    MediaPlayer1->Stop();
    MediaPlayer1->Close();

    MediaPlayer1->EnabledButtons = TButtonSet();

    /* Enable open button again */
    btOpen->Enabled = true;
}

void __fastcall TForm2::FormCreate(TObject *Sender)
{
    /* Disable all buttons */
    MediaPlayer1->AutoEnable = false;
    MediaPlayer1->EnabledButtons = TButtonSet();
}

void __fastcall TForm2::MediaPlayer1PostClick(TObject *Sender,
          TMPBtnType Button)
```

```
{
    if (Button == TMPBtnType::btStop)
        btStop->Click();
}
```

Delphi Examples:

```
{
This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
for Open and Stop are present on the form.
Note: AutoEnable must be set to false to be able to control
the buttons manually.
}

procedure TForm2.btOpenClick(Sender: TObject);
var
  OpenMediaDialog : TOpenDialog;
begin
  OpenMediaDialog := TOpenDialog.Create(Self);
  OpenMediaDialog.Filter := 'All Video Files (*.avi)|*.avi';
  // There are avi files in
  // C:\Documents and Settings\All Users\Documents\RAD
Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
  if OpenMediaDialog.Execute() then
  begin
    { Assign a file to the media player }
    MediaPlayer1.FileName := OpenMediaDialog.FileName;

    { Check if the file exists and is not a directory }
    if (FileExists(OpenMediaDialog.FileName)) and
       (not DirectoryExists(OpenMediaDialog.FileName)) then
    begin
      { Open the files }
      MediaPlayer1.Wait := true;
      MediaPlayer1.Open;
      MediaPlayer1.Play;

      { Override automatic button controlling }
      MediaPlayer1.EnabledButtons :=
        [TMPBtnType.btPause, TMPBtnType.btStop, TMPBtnType.btPlay];

      { Enable  the stop button }
      btStop.Enabled := true;
      btOpen.Enabled := false;
    end;
  end;

  OpenMediaDialog.Free;
end;

procedure TForm2.btStopClick(Sender: TObject);
begin
  { Stop and close the media }
  MediaPlayer1.Stop;
  MediaPlayer1.Close;

  MediaPlayer1.EnabledButtons := [];

  { Enable open button again }
  btOpen.Enabled := true;
end;

procedure TForm2.FormCreate(Sender: TObject);
begin
  { Disable all buttons }
  MediaPlayer1.AutoEnable := false;
  MediaPlayer1.EnabledButtons := [];
```

```
end;

procedure TForm2.MediaPlayer1PostClick(Sender: TObject;
  Button: TMPBtnType);
begin
  if Button = TMPBtnType.btStop then
     btStop.Click;
end;
```

See Also

AutoButtonSet (see page 2312)

DeviceType (see page 2339)

EnabledButtons (see page 2343)

MMNotify (see page 2318)

Mode (see page 2350)

Adding Audio and/or Video Clips to an Application

39.1.2.5.3 TMediaPlayer.AutoOpen Property

Determines if the media player is opened automatically when the application is run.

Pascal

```
property AutoOpen: Boolean;
```

C++

```
__property Boolean AutoOpen;
```

Description

Use AutoOpen to have the media player automatically open at runtime.

If AutoOpen is true, the media player attempts to open the multimedia device specified by the DeviceType (see page 2339) property (or FileName (see page 2347) if DeviceType (see page 2339) is dtAutoSelect) when the form containing the media player component is created at runtime.

If AutoOpen is false, the device must be opened with a call to the Open (see page 2319) method.

If an error occurs when opening the device, an exception of type EMCIDeviceError (see page 2305) exception is raised that contains the error message. Upon completion, a numerical error code is stored in the Error (see page 2346) property, and the corresponding error message is stored in the ErrorMessage (see page 2346) property.

The Wait (see page 2365) property determines whether control is returned to the application before opening the multimedia device. The Notify (see page 2350) property determines whether opening the device generates an OnNotify (see page 2375) event.

See Also

DeviceType (see page 2339)

EMCIDeviceError (see page 2305)

Error (see page 2346)

ErrorMessage (see page 2346)

FileName (see page 2347)

Notify (see page 2350)

OnNotify (see page 2375)

Wait (see page 2365)

Adding Audio and/or Video Clips to an Application

39.1.2.5.4 TMediaPlayer.AutoRewind Property

Determines if the media player control rewinds before playing or recording.

Pascal

```
property AutoRewind: Boolean;
```

C++

```
__property Boolean AutoRewind;
```

Description

If AutoRewind is true and the current position is at the end of the medium, Play (see page 2323) or StartRecording (see page 2329) moves the current position to the beginning of the medium before playing or recording. If AutoRewind is false, the user must click the Prev button or write code to call Previous (see page 2327) to move to the beginning.

If values have been assigned to StartPos (see page 2360) or EndPos (see page 2345) or if the multimedia device uses tracks, AutoRewind has no effect on playing or recording. When calling Play (see page 2323) or StartRecording (see page 2329), the current position remains at the end of the medium.

See Also

Previous (see page 2327)

39.1.2.5.5 TMediaPlayer.Capabilities Property

Determines the capabilities of the open multimedia device.

Pascal

```
property Capabilities: TMPDevCapsSet;
```

C++

```
__property TMPDevCapsSet Capabilities;
```

Description

The various capabilities specified in Capabilities are determined when the device is opened with the Open (see page 2319) method.

Note: Currently, there is no way to check whether a device can step forward or backward. Capabilities includes mpCanStep only if the device type (specified in the DeviceType (see page 2339) property) is Animation, AVI Video, Digital Video, Overlay, or VCR.

See Also

DeviceType (see page 2339)

Open (see page 2319)

39.1.2.5.6 TMediaPlayer.ColoredButtons Property

Determines which buttons on the media player control have color.

Pascal

```
property ColoredButtons: TButtonSet;
```

C++

```
__property TButtonSet ColoredButtons;
```

Description

ColoredButtons controls which buttons on the media player are colored.

If a button is not colored with ColoredButtons, it appears in black-and-white when visible. All media player control buttons are colored by default.

See Also

EnabledButtons (see page 2343)

VisibleButtons (see page 2365)

39.1.2.5.7 TMediaPlayer.Constraints Property

Specifies the size constraints for the control.

Pascal

```
property Constraints;
```

C++

```
__property Constraints;
```

Description

Use Constraints to specify the minimum and maximum width and height of the control. When Constraints contains maximum or minimum values, the control cannot be resized to violate those constraints.

Note: Constraints apply to the height of the form. However, the height of the form depends on how large Windows makes the title bar. For example, Windows XP uses much larger title bars than other windowing systems. To work around this, note the ClientHeight when you design your form and set the constraints in the FormCreate event. In the following example, ClientHeight is represented by x: `Constraints.MinHeight := x + Height - ClientHeight`.

Warning: Do not set up constraints that conflict with the value of the Align or Anchors property. When these properties conflict, the response of the control to resize attempts is not well-defined.

See Also

AutoSize

Align

Anchors

OnConstrainedResize

OnCanResize

39.1.2.5.8 TMediaPlayer.DeviceID Property

Specifies the device ID for the currently open multimedia device.

Pascal

```
property DeviceID: Word;
```

C++

```
__property Word DeviceID;
```

Description

The value of DeviceID is determined when a device is opened with the Open (see page 2319) method. If no device is open, DeviceID is 0.

See Also

Open (see page 2319)

39.1.2.5.9 TMediaPlayer.DeviceType Property

Specifies a multimedia device type to open the media player.

Pascal

```
property DeviceType: TMPDeviceTypes;
```

C++

```
__property TMPDeviceTypes DeviceType;
```

Description

DeviceType specifies a multimedia device type to open with the Open (see page 2319) method. The default is dtAutoSelect.

If DeviceType is dtAutoSelect, the device type is determined by the file extension specified in the FileName (see page 2347) property. If no device type is associated with the extension, the correct device type must be explicitly specified by setting DeviceType to a value other than dtAutoSelect.

A multimedia device is typically associated with an appropriate file name extension when the device is installed. Associations are specified in the registry or SYSTEM.INI file. See the documentation for the specific device for instructions about how to associate file name extensions with the device.

See Also

FileName (see page 2347)

Open (see page 2319)

Adding Audio and/or Video Clips to an Application

39.1.2.5.10 TMediaPlayer.Display Property

Specifies the display window for a multimedia device that uses a window for output.

Pascal

```
property Display: TWinControl;
```

C++

```
__property TWinControl * Display;
```

Description

To use Display, assign the name of a windowed control such as a form or panel to Display to display output in that control.

The default value of Display is nil (Delphi) or NULL (C++), meaning that the device creates its own window to display output.

Also, by freeing the control assigned to Display after the device has been opened, video output will be in its own default window.

Examples of multimedia devices that use a window to display output are Animation, AVI Video, Digital Video, Overlay, and VCR.

See Also

DisplayRect (see page 2340)

Adding Audio and/or Video Clips to an Application

39.1.2.5.11 TMediaPlayer.DisplayRect Property

Specifies a rectangular area in the control specified by the Display (see page 2339) property that is used to display output from a multimedia device.

Pascal

```
property DisplayRect: TRect;
```

C++

```
__property TRect DisplayRect;
```

Description

To display output in a specific rectangle area on a form assign a TRect record to DisplayRect. The Rect function can be used to create a TRect record.

Media that use a rectangle to display output usually perform best if the default DisplayRect size is used. To set DisplayRect to the default size, position the rectangle in the upper left corner and use 0, 0 for the lower right corner.

DisplayRect is ignored if Display (see page 2339) is nil (Delphi) or NULL (C++).

Examples of multimedia devices that use a window to display output are Animation, AVI Video, Digital Video, Overlay, and VCR.

Note: DisplayRect can be set only after the media device is opened.

See Also

Display (see page 2339)

Open (see page 2319)

39.1.2.5.12 TMediaPlayer.DoubleBuffered Property

Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.

Pascal

```
property DoubleBuffered;
```

C++

```
__property DoubleBuffered;
```

Description

When DoubleBuffered is false, the windowed control paints itself directly to the window. When DoubleBuffered is true, the windowed control paints itself to an in-memory bitmap that is then used to paint the window. Double buffering reduces the amount of flicker when the control repaints, but is more memory intensive.

When a windowed control is a dock site and has an associated dock manager, it must be double-buffered.

Note: Some controls, such as TRichEdit, can't paint themselves into a bitmap. For such controls, DoubleBuffered must be set to false.

C++ Examples:

```
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
dropped label.
}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Not');
  ListBox1.Items.Add('In');
  ListBox1.Items.Add('Alphabetical');
  ListBox1.Items.Add('Order');
end;

// This OnDragOver event handler permits the list box to
// accept a dropped label:
```

```
procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

// This OnDragDrop event handler implements the drop behavior.

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
    with Sender as TListBox do
    begin
      Font := (Source as TLabel).Font;
      Color := (Source as TLabel).Color;
      DoubleBuffered := true;
      Color := clWindow;
    end;
  end;
end;
```

See Also

DockSite

DockManager

PaintWindow

39.1.2.5.13 TMediaPlayer.Enabled Property

Controls whether the control responds to mouse, keyboard, and timer events.

Pascal

```
property Enabled;
```

C++

```
__property Enabled;
```

Description

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to False. Some disabled controls appear dimmed (for example: buttons, check boxes, labels), while others (container controls) simply lose their functionality without changing their appearance. If Enabled is set to False, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to True.

Note: This property applies to all TControl descendants.

See Also

Visible

DisableControls

EnableControls

Enabling Mouse

39.1.2.5.14 TMediaPlayer.EnabledButtons Property

Determines which buttons on the media player are usable.

Pascal

```
property EnabledButtons: TButtonSet;
```

C++

```
__property TButtonSet EnabledButtons;
```

Description

EnabledButtons controls which buttons on the media player are enabled and usable.

An enabled button is colored and usable. A disabled button is dimmed and not usable. If a button is not enabled with EnabledButtons, it is disabled. By default, all buttons are enabled.

If the AutoEnable (see page 2333) property is true, AutoEnable (see page 2333) supersedes EnabledButtons. The buttons automatically enabled or disabled by the media player override any buttons enabled or disabled with the EnabledButtons property.

C++ Examples:

```
//
//This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
//for Open and Stop are present on the form.
//Note: AutoEnable must be set to false to be able to control
//the buttons manually.
//
void __fastcall TForm2::btOpenClick(TObject *Sender)
{
    TOpenDialog* OpenMediaDialog;

    OpenMediaDialog = new TOpenDialog(this);
    OpenMediaDialog->Filter = "All Video Files (*.avi)|*.avi";
  // There are avi files in
  // C:\Documents and Settings\All Users\Documents\RAD
Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
    if (OpenMediaDialog->Execute())
    {
        /* Assign a file to the media player */
        MediaPlayer1->FileName = OpenMediaDialog->FileName;

        /* Check if the file exists and is not a directory */
        if ((FileExists(OpenMediaDialog->FileName)) &&
           (!DirectoryExists(OpenMediaDialog->FileName)))
        {
            /* Open the files */
            MediaPlayer1->Wait = true;
            MediaPlayer1->Open();
            MediaPlayer1->Play();

            /* Override automatic button controlling */
            MediaPlayer1->EnabledButtons =
                TButtonSet() <<
                TMPBtnType::btPause <<
                TMPBtnType::btStop <<
                TMPBtnType::btPlay;


            /* Enable  the stop button */
            btStop->Enabled = true;
            btOpen->Enabled = false;
        }
```

```
    }

    delete OpenMediaDialog;
}

void __fastcall TForm2::btStopClick(TObject *Sender)
{
    /* Stop and close the media */
    MediaPlayer1->Stop();
    MediaPlayer1->Close();

    MediaPlayer1->EnabledButtons = TButtonSet();

    /* Enable open button again */
    btOpen->Enabled = true;
}

void __fastcall TForm2::FormCreate(TObject *Sender)
{
    /* Disable all buttons */
    MediaPlayer1->AutoEnable = false;
    MediaPlayer1->EnabledButtons = TButtonSet();
}

void __fastcall TForm2::MediaPlayer1PostClick(TObject *Sender,
          TMPBtnType Button)
{
    if (Button == TMPBtnType::btStop)
        btStop->Click();
}
```

Delphi Examples:

```
{
This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
for Open and Stop are present on the form.
Note: AutoEnable must be set to false to be able to control
the buttons manually.
}

procedure TForm2.btOpenClick(Sender: TObject);
var
  OpenMediaDialog : TOpenDialog;
begin
  OpenMediaDialog := TOpenDialog.Create(Self);
  OpenMediaDialog.Filter := 'All Video Files (*.avi)|*.avi';
  // There are avi files in
  // C:\Documents and Settings\All Users\Documents\RAD
Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
  if OpenMediaDialog.Execute() then
  begin
    { Assign a file to the media player }
    MediaPlayer1.FileName := OpenMediaDialog.FileName;

    { Check if the file exists and is not a directory }
    if (FileExists(OpenMediaDialog.FileName)) and
       (not DirectoryExists(OpenMediaDialog.FileName)) then
    begin
      { Open the files }
      MediaPlayer1.Wait := true;
      MediaPlayer1.Open;
      MediaPlayer1.Play;

      { Override automatic button controlling }
      MediaPlayer1.EnabledButtons :=
        [TMPBtnType.btPause, TMPBtnType.btStop, TMPBtnType.btPlay];
```

```
        { Enable  the stop button }
        btStop.Enabled := true;
        btOpen.Enabled := false;
      end;
    end;

    OpenMediaDialog.Free;
  end;

  procedure TForm2.btStopClick(Sender: TObject);
  begin
    { Stop and close the media }
    MediaPlayer1.Stop;
    MediaPlayer1.Close;

    MediaPlayer1.EnabledButtons := [];

    { Enable open button again }
    btOpen.Enabled := true;
  end;

  procedure TForm2.FormCreate(Sender: TObject);
  begin
    { Disable all buttons }
    MediaPlayer1.AutoEnable := false;
    MediaPlayer1.EnabledButtons := [];
  end;

  procedure TForm2.MediaPlayer1PostClick(Sender: TObject;
    Button: TMPBtnType);
  begin
    if Button = TMPBtnType.btStop then
       btStop.Click;
  end;
```

See Also

ColoredButtons (see page 2337)

VisibleButtons (see page 2365)

Adding Audio and/or Video Clips to an Application

39.1.2.5.15 TMediaPlayer.EndPos Property

Specifies the position within the currently loaded medium at which to stop playing or recording.

Pascal

```
property EndPos: Longint;
```

C++

```
__property Longint EndPos;
```

Description

EndPos is specified using the current time format, which is specified in the TimeFormat (see page 2362) property.

The EndPos property affects only the next Play (see page 2323) or StartRecording (see page 2329) method called after setting EndPos. Reset EndPos to affect any subsequent calls to Play (see page 2323) or StartRecording (see page 2329).

See Also

Play (see page 2323)

StartRecording (see page 2329)

TimeFormat (see page 2362)

39.1.2.5.16 TMediaPlayer.Error Property

Specifies the MCI error code returned by the media control method.

Pascal

```
property Error: Longint;
```

C++

```
__property Longint Error;
```

Description

Error specifies the MCI error code returned by the most recent media control method (Back (see page 2313), Close (see page 2314), Eject (see page 2315), Next (see page 2318), Open (see page 2319), Pause (see page 2322), PauseOnly (see page 2323), Play (see page 2323), Previous (see page 2327), StartRecording (see page 2329), Resume (see page 2327), Rewind (see page 2328), Step (see page 2330), or Stop (see page 2330)). The ErrorMessage (see page 2346) property describes the Error.

The error code returned by media control methods are the same error code returned by the mciSendCommand function. The message describing the error code is stored in the ErrorMessage (see page 2346) property.

The value of Error is zero if the most recent media control method didn't cause an error. If a method results in an error, a value other than zero is stored in Error. If the error occurs during the opening of the device, an EMCIDeviceError (see page 2305) exception occurs.

See Also

EMCIDeviceError (see page 2305)

ErrorMessage (see page 2346)

Play (see page 2323)

StartRecording (see page 2329)

TimeFormat (see page 2362)

39.1.2.5.17 TMediaPlayer.ErrorMessage Property

Describes the error code stored in the Error (see page 2346) property.

Pascal

```
property ErrorMessage: string;
```

C++

```
__property UnicodeString ErrorMessage;
```

Description

ErrorMessage specifies the error message that describes the error code returned from the most recent media control method (Back (see page 2313), Close (see page 2314), Eject (see page 2315), Next (see page 2318), Open (see page 2319), Pause (see page 2322), PauseOnly (see page 2323), Play (see page 2323), Previous (see page 2327), StartRecording (see page 2329), Resume (see page 2327), Rewind (see page 2328), Step (see page 2330), or Stop (see page 2330)).

The error code described by the message is stored in the Error (see page 2346) property.

See Also

Error (see page 2346)

39.1.2.5.18 TMediaPlayer.FileName Property

Specifies the opened or saved media file.

Pascal

```
property FileName: string;
```

C++

```
__property UnicodeString FileName;
```

Description

FileName specifies the media file to be opened by the Open (see page 2319) method, or the file to save by the Save (see page 2328) method.

At design time, use a file open dialog box to specify the FileName property by clicking the ellipsis button (...) in the Object Inspector.

C++ Examples:

```
//
//This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
//for Open and Stop are present on the form.
//Note: AutoEnable must be set to false to be able to control
//the buttons manually.
//
void __fastcall TForm2::btOpenClick(TObject *Sender)
{
    TOpenDialog* OpenMediaDialog;

    OpenMediaDialog = new TOpenDialog(this);
    OpenMediaDialog->Filter = "All Video Files (*.avi)|*.avi";
  // There are avi files in
  // C:\Documents and Settings\All Users\Documents\RAD
Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
    if (OpenMediaDialog->Execute())
    {
        /* Assign a file to the media player */
        MediaPlayer1->FileName = OpenMediaDialog->FileName;

        /* Check if the file exists and is not a directory */
        if ((FileExists(OpenMediaDialog->FileName)) &&
           (!DirectoryExists(OpenMediaDialog->FileName)))
        {
            /* Open the files */
            MediaPlayer1->Wait = true;
            MediaPlayer1->Open();
            MediaPlayer1->Play();

            /* Override automatic button controlling */
            MediaPlayer1->EnabledButtons =
                TButtonSet() <<
                TMPBtnType::btPause <<
                TMPBtnType::btStop <<
                TMPBtnType::btPlay;


            /* Enable  the stop button */
            btStop->Enabled = true;
            btOpen->Enabled = false;
```

```
        }
    }

    delete OpenMediaDialog;
}

void __fastcall TForm2::btStopClick(TObject *Sender)
{
    /* Stop and close the media */
    MediaPlayer1->Stop();
    MediaPlayer1->Close();

    MediaPlayer1->EnabledButtons = TButtonSet();

    /* Enable open button again */
    btOpen->Enabled = true;
}

void __fastcall TForm2::FormCreate(TObject *Sender)
{
    /* Disable all buttons */
    MediaPlayer1->AutoEnable = false;
    MediaPlayer1->EnabledButtons = TButtonSet();
}

void __fastcall TForm2::MediaPlayer1PostClick(TObject *Sender,
          TMPBtnType Button)
{
    if (Button == TMPBtnType::btStop)
        btStop->Click();
}
```

Delphi Examples:

```
{
This example demonstrates the use of TMediaPlayer component. Assuming 2 buttons
for Open and Stop are present on the form.
Note: AutoEnable must be set to false to be able to control
the buttons manually.
}

procedure TForm2.btOpenClick(Sender: TObject);
var
  OpenMediaDialog : TOpenDialog;
begin
  OpenMediaDialog := TOpenDialog.Create(Self);
  OpenMediaDialog.Filter := 'All Video Files (*.avi)|*.avi';
  // There are avi files in
  // C:\Documents and Settings\All Users\Documents\RAD
Studio\6.0\Demos\DelphiWin32\VCLWin32\CoolStuff
  if OpenMediaDialog.Execute() then
  begin
    { Assign a file to the media player }
    MediaPlayer1.FileName := OpenMediaDialog.FileName;

    { Check if the file exists and is not a directory }
    if (FileExists(OpenMediaDialog.FileName)) and
       (not DirectoryExists(OpenMediaDialog.FileName)) then
    begin
      { Open the files }
      MediaPlayer1.Wait := true;
      MediaPlayer1.Open;
      MediaPlayer1.Play;

      { Override automatic button controlling }
      MediaPlayer1.EnabledButtons :=
        [TMPBtnType.btPause, TMPBtnType.btStop, TMPBtnType.btPlay];
```

```
        { Enable  the stop button }
        btStop.Enabled := true;
        btOpen.Enabled := false;
      end;
    end;

    OpenMediaDialog.Free;
  end;

  procedure TForm2.btStopClick(Sender: TObject);
  begin
    { Stop and close the media }
    MediaPlayer1.Stop;
    MediaPlayer1.Close;

    MediaPlayer1.EnabledButtons := [];

    { Enable open button again }
    btOpen.Enabled := true;
  end;

  procedure TForm2.FormCreate(Sender: TObject);
  begin
    { Disable all buttons }
    MediaPlayer1.AutoEnable := false;
    MediaPlayer1.EnabledButtons := [];
  end;

  procedure TForm2.MediaPlayer1PostClick(Sender: TObject;
    Button: TMPBtnType);
  begin
    if Button = TMPBtnType.btStop then
       btStop.Click;
  end;
```

See Also

Open (see page 2319)

Save (see page 2328)

Adding Audio and/or Video Clips to an Application

39.1.2.5.19 TMediaPlayer.Frames Property

Specifies the number of frames moved forward or backward.

Pascal

```
property Frames: Longint;
```

C++

```
__property Longint Frames;
```

Description

Frames specifies the number of frames the Step (see page 2330) method steps forward or the Back (see page 2313) method steps backward. Frames defaults to ten percent of the length of the currently loaded medium, which is specified by the Length (see page 2350) property.

The definition of a frame varies by multimedia device. For display media, a frame is one still image.

See Also

Back (see page 2313)

Length (see page 2350)

Step (see page 2330)

39.1.2.5.20 TMediaPlayer.Length Property

Specifies the length of the medium in the open multimedia device.

Pascal

```
property Length: Longint;
```

C++

```
__property Longint Length;
```

Description

Length is specified using the current time format, which is specified by the TimeFormat (see page 2362) property.

See Also

TimeFormat (see page 2362)

39.1.2.5.21 TMediaPlayer.Mode Property

Indicates the state of the currently open multimedia device.

Pascal

```
property Mode: TMPModes;
```

C++

```
__property TMPModes Mode;
```

Description

Mode specifies the current state or mode of the currently open multimedia device.

The TMPModes (see page 2380) type defines the modes for a multimedia device used with a TMediaPlayer (see page 2305).

See Also

OnNotify (see page 2375)

39.1.2.5.22 TMediaPlayer.Notify Property

Determines whether an OnNotify (see page 2375) event is generated.

Pascal

```
property Notify: Boolean;
```

C++

```
__property Boolean Notify;
```

Description

Notify determines whether the next call to a media control method (Back (see page 2313), Close (see page 2314), Eject (see page 2315), Next (see page 2318), Open (see page 2319), Pause (see page 2322), PauseOnly (see page 2323), Play (see page 2323), Previous (see page 2327), StartRecording (see page 2329), Resume (see page 2327), Rewind (see page 2328), Step (see page 2330), or Stop (see page 2330)) generates an OnNotify (see page 2375) event when

the method has completed.

If Notify is true, the next media control method generates OnNotify (see page 2375) event upon completion and stores the notification message in the NotifyValue (see page 2351) property. If Notify is false, the method does not generate an OnNotify (see page 2375) event and NotifyValue (see page 2351) remains unchanged.

Notify affects only the next call to a media control method. After an OnNotify (see page 2375) event, Notify must be reset to affect any subsequent media control methods.

By default, Play (see page 2323) and StartRecording (see page 2329) function as if Notify is true. Set Notify to false before calling Play (see page 2323) or StartRecording (see page 2329) to prevent an OnNotify (see page 2375) event from being generated when playing or recording has finished. By default, all other media control methods function as if Notify is false.

Tip: Set Notify to true if the next media control is expected to take a long time, so the application is notified when the media control method has completed. If Notify is set to true, it is recommended to set Wait (see page 2365) to false so that control returns to the application before the media control method is finished.

Note: When trying to resume a device that doesn't support Resume (see page 2327), the device is resumed as if the Play (see page 2323) method was called. If Notify was assigned true before calling Resume (see page 2327) (or any other media control method), Notify doesn't affect the call to Resume (see page 2327). Resume (see page 2327) does not generate an OnNotify (see page 2375) event upon completion, and NotifyValue (see page 2351) remains unchanged.

See Also

NotifyValue (see page 2351)

OnNotify (see page 2375)

Resume (see page 2327)

Wait (see page 2365)

39.1.2.5.23 TMediaPlayer.NotifyValue Property

Indicates the result of the last media control method that requested a notification.

Pascal

```
property NotifyValue: TMPNotifyValues;
```

C++

```
__property TMPNotifyValues NotifyValue;
```

Description

NotifyValue reports the result of the last media control method (Back (see page 2313), Close (see page 2314), Eject (see page 2315), Next (see page 2318), Open (see page 2319), Pause (see page 2322), PauseOnly (see page 2323), Play (see page 2323), Previous (see page 2327), StartRecording (see page 2329), Resume (see page 2327), Rewind (see page 2328), Step (see page 2330), or Stop (see page 2330)) that requested a notification.

To request notification, set Notify (see page 2350) to true before calling a media control method.

See Also

OnNotify (see page 2375)

Notify (see page 2350)

39.1.2.5.24 TMediaPlayer.ParentDoubleBuffered Property

ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.

Pascal

```
property ParentDoubleBuffered;
```

C++

```
__property ParentDoubleBuffered;
```

Description

ParentDoubleBuffered is a boolean indicating that the DoubleBuffered property of this component is overridden by the value of the parent's DoubleBuffered property.

39.1.2.5.25 TMediaPlayer.ParentShowHint Property

Determines where a control looks to find out if its Help Hint should be shown.

Pascal

```
property ParentShowHint;
```

C++

```
__property ParentShowHint;
```

Description

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the control uses the ShowHint property value of its parent. If ParentShowHint is false, the control uses the value of its own ShowHint property.

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note: Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See Also

Hint

Parent

ParentColor

ParentFont

ShowHint

39.1.2.5.26 TMediaPlayer.PopupMenu Property

Identifies the pop-up menu associated with the control.

Pascal

```
property PopupMenu;
```

C++

```
__property PopupMenu;
```

Description

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the control and clicks the right mouse button. If the TPopupMenu's AutoPopup property is true, the pop-up menu appears automatically. If the menu's AutoPopup property is false, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

C++ Examples:

```
/*
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  PopupMenu1->AutoPopup = true;
  Edit1->PopupMenu = PopupMenu1;
  Edit2->PopupMenu = PopupMenu1;
}

/*
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
*/
void __fastcall TForm1::Copy1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CopyToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Cut1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CutToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Paste1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
```

```
    {
      if (pComponent->ClassNameIs("TEdit"))
        (dynamic_cast<TEdit *>(pComponent))->PasteFromClipboard();
      else
        MessageBeep(0);
    }
    else
      MessageBeep(0);
}
/*
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
    {
        switch (ClickedItem->Tag)
        {
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
            }
        }
    }
}

/*
To edit or destroy an item, grab its pointer via the Items
property.
```

```
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}

void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  PopupMenu1.AutoPopup := True;
  Edit1.PopupMenu := PopupMenu1;
  Edit2.PopupMenu := PopupMenu1;
end;

{
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
}
procedure TForm1.Copy1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
   Edit1.CopyToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CopyToClipboard
  else
    Beep;
end;

procedure TForm1.Cut1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
```

```
    Edit1.CutToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CutToClipboard
  else
    Beep;
end;

procedure TForm1.Paste1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.PasteFromClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.PasteFromClipboard
  else
    Beep;
end;
{
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
}

type
  TForm1 = class(TForm)
    AddButton: TButton;
    EditButton: TButton;
    DestroyButton: TButton;
    PopupMenu1: TPopupMenu;
    procedure AddButtonClick(Sender: TObject);
    procedure EditButtonClick(Sender: TObject);
    procedure DestroyButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    procedure PopupMenuItemsClick(Sender: TObject);

  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.AddButtonClick(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
    // The owner (PopupMenu1) will cleanup this menu item.
  NewItem := TMenuItem.Create(PopupMenu1); // create the new item
  index := PopupMenu1.Items.Count;
  PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
  NewItem.Caption := 'Menu Item ' + IntToStr(index);
  NewItem.Tag := index;
  NewItem.OnClick :=
    PopupMenuItemsClick; // assign it an event handler
end;
```

```
procedure TForm1.PopupMenuItemsClick(Sender: TObject);
begin
  with Sender as TMenuItem do
  begin
    case Tag of
      0:  ShowMessage('first item clicked');
      1:  ShowMessage('second item clicked');
      2:  ShowMessage('third item clicked');
      3:  ShowMessage('fourth item clicked');
    end;
  end;
end;

{
To edit or destroy an item, grab its pointer via the Items
property.
}
procedure TForm1.EditButtonClick(Sender: TObject);
var
  ItemToEdit: TMenuItem;
begin
  ItemToEdit := PopupMenu.Items[1];
  ItemToEdit.Caption := 'Changed Caption';
end;

procedure TForm1.DestroyButtonClick(Sender: TObject);
var
  ItemToDelete: TMenuItem;
begin
  ItemToDelete := PopupMenu.Items[2];
  ItemToDelete.Free;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
  for index := 0 to 3 do
  begin
      // The owner (PopupMenu1) will cleanup this menu item.
    NewItem := TMenuItem.Create(PopupMenu1); // create the new item
    PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
    NewItem.Caption := 'Menu Item ' + IntToStr(index);
    NewItem.Tag := index;
    NewItem.OnClick :=
      PopupMenuItemsClick; // assign it an event handler
  end;
end;
```

See Also

AutoPopup (see page 1437)

Popup (see page 1434)

OnContextPopup

39.1.2.5.27 TMediaPlayer.Position Property

Specifies the current position within the currently loaded medium.

Pascal

```
property Position: Longint;
```

C++

```
__property Longint Position;
```

Description

The value of Position is specified according to the current time format, which is specified in the TimeFormat (see page 2362) property.

Position defaults to the beginning of the medium. If the medium supports multiple tracks, Position defaults to the beginning of the first track.

See Also

TimeFormat (see page 2362)

39.1.2.5.28 TMediaPlayer.Shareable Property

Determines whether more than one application can share a multimedia device.

Pascal

```
property Shareable: Boolean;
```

C++

```
__property Boolean Shareable;
```

Description

If Shareable is false, no other components or applications can access the device. If Shareable is true, more than one component or application can access the device. Shareable defaults to false.

Be sure to set Shareable before opening a device.

Some devices are not shareable. If Shareable is set to true for a device that isn't shareable, the Open (see page 2319) method fails and the error code is returned to the Error (see page 2346) property.

See Also

Error (see page 2346)

Open (see page 2319)

39.1.2.5.29 TMediaPlayer.ShowHint Property

Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.

Pascal

```
property ShowHint;
```

C++

```
__property ShowHint;
```

Description

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either

the control's own ShowHint property must be true, or

the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

C++ Examples:

```
/*
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  Edit1->Hint = "Enter your name";
  Edit1->ShowHint = true;

  char string[10];
  char index[3];
  for(int i = 1; i <= 10; i++)
  {
    strcpy(string, "Item");
    itoa(i, index, 10);
    strcat(string, index);
    ListBox1->Items->Add(string);
  }
  ListBox1->Hint = "Select an item";
  ListBox1->ShowHint = true;
  Application->ShowHint = true;
}
/*
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
*/
```

Delphi Examples:

```
{
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
  Edit1.Hint := 'Enter your name';
  Edit1.ShowHint := True;
  with ListBox1 do
  begin
    for I := 1 to 10 do
      Items.Add('Item ' + IntToStr(I));
    Hint := 'Select an item';
    ShowHint := True;
  end;
  Application.ShowHint := True;
end;
{
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
}
```

See Also

Hint

ParentShowHint

ShowHint

Displaying Fly-over Help

39.1.2.5.30 TMediaPlayer.Start Property

Specifies the starting position within the currently loaded medium.

Pascal

```
property Start: Longint;
```

C++

```
__property Longint Start;
```

Description

Start is the beginning of the medium for devices that don't use tracks, or the beginning of the first track for devices that use tracks. Start is defined when a multimedia device is opened with the Open (see page 2319) method. Start is specified according to the current time format, which is stored in the TimeFormat (see page 2362) property. Start is read-only at runtime and is unavailable at design time.

See Also

Open (see page 2319)

TimeFormat (see page 2362)

39.1.2.5.31 TMediaPlayer.StartPos Property

Specifies the position within the currently loaded medium from which to begin playing or recording.

Pascal

```
property StartPos: Longint;
```

C++

```
__property Longint StartPos;
```

Description

StartPos is specified using the current time format, which is specified in the TimeFormat (see page 2362) property.

The StartPos property affects only the next Play (see page 2323) or StartRecording (see page 2329) method called after setting StartPos. Reset must be set to StartPos to affect any subsequent calls to Play (see page 2323) or StartRecording (see page 2329).

StartPos does not affect the current position of the medium (specified in the Position (see page 2357) property) until the next Play (see page 2323) or StartRecording (see page 2329) method is called.

See Also

Play (see page 2323)

StartRecording (see page 2329)

TimeFormat (see page 2362)

39.1.2.5.32 TMediaPlayer.TabOrder Property

Indicates the position of the control in its parent's tab order.

Pascal

```
property TabOrder;
```

C++

```
__property TabOrder;
```

Description

TabOrder is the order in which child windows are visited when the user presses the Tab key. The control with the TabOrder value of 0 is the control that has the focus when the form first appears.

Initially, the tab order is always the order in which the controls were added to the form. The first control added to the form has a TabOrder value of 0, the second is 1, the third is 2, and so on. Change this by changing the TabOrder property.

Each control has a unique tab-order value within its parent. If you change the TabOrder property value of one control to be the same as the value of a second control, the TabOrder value for all the other controls changes. For example, suppose a control is sixth in the tab order. If you change the control's TabOrder property value to 3 (making the control fourth in the tab order), the control that was originally fourth in the tab order now becomes fifth, and the control that was fifth becomes sixth.

Assigning TabOrder a value greater than the number of controls contained in the parent control moves the control to the end of the tab order. The control does not take on the assigned value of TabOrder, but instead is given the number that assures the control is the last in the tab order.

Note: TabOrder is meaningful only if the TabStop property is true and if the control has a parent. (The TabOrder property of a form is not used unless the form is the child of another form.) A control with a TabOrder of -1 has no parent, and therefore cannot be reached by pressing the Tab key. To remove a parented control from the Tab order, set its TabStop property to false.

See Also

FindNextControl

TabStop

39.1.2.5.33 TMediaPlayer.TabStop Property

Determines if the user can tab to a control.

Pascal

```
property TabStop;
```

C++

```
__property TabStop;
```

Description

Use the TabStop to allow or disallow access to the control using the Tab key.

If TabStop is true, the control is in the tab order. If TabStop is false, the control is not in the tab order and the user can't press the Tab key to move to the control.

Note: TabStop is not meaningful for a form unless the form assigns another form to be its parent.

See Also

TabOrder

39.1.2.5.34 TMediaPlayer.TimeFormat Property

Determines the format used to specify position information.

Pascal

```
property TimeFormat: TMPTimeFormats;
```

C++

```
__property TMPTimeFormats TimeFormat;
```

Description

TimeFormat determines how the StartPos (see page 2360), Length (see page 2350), Position (see page 2357), Start (see page 2360), and EndPos (see page 2345) properties are interpreted. For example, if Position (see page 2357) is 180 and TimeFormat is tfMilliseconds, the current position is 180 milliseconds into the medium. If Position (see page 2357) is 180 and TimeFormat is tfMSF, the current position is 180 minutes into the medium.

Not all formats are supported by every device. When trying to set an unsupported format, the assignment is ignored.

The current timing information is always passed in a 4-byte integer. In some formats, the timing information returned is not really one integer, but single bytes of information packed in the long integer.

See Also

EndPos (see page 2345)

Length (see page 2350)

Position (see page 2357)

Start (see page 2360)

StartPos (see page 2360)

39.1.2.5.35 TMediaPlayer.TrackLength Property

Reports the length of the track.

Pascal

```
property TrackLength [TrackNum: Integer]: Longint;
```

C++

```
__property Longint TrackLength[int TrackNum];
```

Description

TrackLength reports the length of the track specified by the TrackNum index.

The value of TrackLength is specified according to the current time format, which is specified in the TimeFormat (see page 2362) property.

See Also

TimeFormat (see page 2362)

Tracks (see page 2363)

39.1.2.5.36 TMediaPlayer.TrackPosition Property

Reports the starting position of the track.

Pascal

```
property TrackPosition [TrackNum: Integer]: Longint;
```

C++

```
__property Longint TrackPosition[int TrackNum];
```

Description

TrackPosition reports the starting position of the track specified by the TrackNum index.

The value of TrackPosition is specified according to the current time format, which is specified in the TimeFormat (see page 2362) property.

See Also

TimeFormat (see page 2362)

Tracks (see page 2363)

39.1.2.5.37 TMediaPlayer.Tracks Property

Specifies the number of playable tracks on the open multimedia device.

Pascal

```
property Tracks: Longint;
```

C++

```
__property Longint Tracks;
```

Description

Tracks indicates how many of playable tracks are available on the open multimedia device.

Tracks is undefined for devices that don't use tracks.

See Also

TrackLength (see page 2362)

TrackPosition (see page 2362)

Adding Audio and/or Video Clips to an Application

39.1.2.5.38 TMediaPlayer.Visible Property

Determines whether the component appears onscreen.

Pascal

```
property Visible;
```

C++

```
__property Visible;
```

Description

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

Calling the Show method sets the control's Visible property to true. Calling the Hide method sets it to false.

C++ Examples:

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
    pPage->PageControl = PageControl1;
    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this); // The owner will clean this up.
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
    begin
      PageControl := PageControl1;
      Caption := 'TabSheet' + IntToStr(i);
      with TEdit.Create(Self) do
      begin
        Parent := PageControl1.Pages[i];
        Left := Random(PageControl1.ActivePage.ClientWidth - Width);
        Top := Random(PageControl1.ActivePage.ClientHeight - Height);
      end;
    end;
```

```
    PageControl1Change(Sender);
end;
```

See Also

Enabled

Hide

Show

VisibleChanging

39.1.2.5.39 TMediaPlayer.VisibleButtons Property

Indicates which of the buttons on the media player are visible.

Pascal

```
property VisibleButtons: TButtonSet;
```

C++

```
__property TButtonSet VisibleButtons;
```

Description

VisibleButtons determines which buttons on the media player that are visible. If a button is not made visible with VisibleButtons, it does not appear on the media player control. By default, all buttons are visible when a media player component is added to a form.

See Also

ColoredButtons (see page 2337)

EnabledButtons (see page 2343)

39.1.2.5.40 TMediaPlayer.Wait Property

Determines whether a media control method returns control to the application only after it has been completed.

Pascal

```
property Wait: Boolean;
```

C++

```
__property Boolean Wait;
```

Description

Wait determines whether a media control method (Back (see page 2313), Close (see page 2314), Eject (see page 2315), Next (see page 2318), Open (see page 2319), Pause (see page 2322), PauseOnly (see page 2323), Play (see page 2323), Previous (see page 2327), StartRecording (see page 2329), Resume (see page 2327), Rewind (see page 2328), Step (see page 2330), or Stop (see page 2330)) returns control to the application only after it has been completed.

Wait is unavailable at design time.

If Wait is true, the media player component waits until the next media control method has completed before returning control to the application. If Wait is false, the application won't wait for the next media control method to finish before continuing.

Wait affects only the next media control method called after setting Wait. Wait must be reset to affect any subsequent call to a media control method.

By default, Play (see page 2323) and StartRecording (see page 2329) function as if Wait is false. Wait must be set to true

before calling Play (see page 2323) or StartRecording (see page 2329) to prevent control from returning to the application before playing or recording has finished. By default, all other media control methods function as if Wait is true.

Note: Wait is usually set to false only if the next media control is expected to take a long time, so that the application can execute other code before the media control method has completed. If Wait is set to false, you should set Notify (see page 2350) to true so the application is notified when the media control method completes.

See Also

Notify (see page 2350)

39.1.2.6 TMediaPlayer Events

The events of the TMediaPlayer class are listed here.

Published Events

	Name	Description
	OnClick (see page 2366)	Occurs when the user presses and releases the mouse button while the mouse pointer is over one of the control buttons, or when the user presses Spacebar while the media player control has focus.
	OnContextPopup (see page 2367)	Occurs when the user right-clicks the control or otherwise invokes the popup menu (such as using the keyboard).
	OnEnter (see page 2368)	Occurs when a control receives the input focus.
	OnExit (see page 2370)	Occurs when the input focus shifts away from one control to another.
	OnMouseEnter (see page 2374)	Occurs when the user moves the mouse into a control.
	OnMouseLeave (see page 2374)	Occurs when the user moves the mouse outside of a control.
	OnNotify (see page 2375)	Occurs upon the completion of a media control method.
	OnPostClick (see page 2375)	Occurs after the OnClick (see page 2366) event handler is called.

39.1.2.6.1 TMediaPlayer.OnClick Event

Occurs when the user presses and releases the mouse button while the mouse pointer is over one of the control buttons, or when the user presses Spacebar while the media player control has focus.

Pascal

```
property OnClick: EMPNotify;
```

C++

```
__property EMPNotify OnClick;
```

Description

When the media player control has focus and the Spacebar is pressed, the user can use the Left Arrow or Right Arrow keys to select which control button to click.

The EMPNotify (see page 2384) type is a method pointer that is called when an OnClick event for a TMediaPlayer (see page 2305) components occurs. The Button argument can be one of the following values: btBack, btEject, btNext, btPause, btPlay, btPrev, btRecord, btStep, or btStop.

The default value of the event handler's DoDefault argument is true. If DoDefault is true, the media player control calls the method that corresponds to the clicked button. For example, if the user clicks the Play (see page 2323) button (btPlay), the Play (see page 2323) method is called.

If DoDefault is false, the user must write code that executes when a media player control button is clicked in the OnClick event handler. The following table lists the default methods corresponding to the media player control buttons:

Control button	Button value	Method called
Play (see page 2323)	btPlay	Play (see page 2323)
Record	btRecord	StartRecording (see page 2329)
Stop (see page 2330)	btStop	Stop (see page 2330)
Next (see page 2318)	btNext	Next (see page 2318)
Prev	btPrev	Previous (see page 2327)
Step (see page 2330)	btStep	Step (see page 2330)
Back (see page 2313)	btBack	Back (see page 2313)
Pause (see page 2322)	btPause	Pause (see page 2322)
Eject (see page 2315)	btEject	Eject (see page 2315)

See Also

Click (see page 2313)

39.1.2.6.2 TMediaPlayer.OnContextPopup Event

Occurs when the user right-clicks the control or otherwise invokes the popup menu (such as using the keyboard).

Pascal

```
property OnContextPopup;
```

C++

```
__property OnContextPopup;
```

Description

The OnContextPopup handler is called when the user uses the mouse or keyboard to request a popup menu. The OnContextPopup event is generated by a WM_CONTEXTMENU message, which is itself generated by the user clicking the right mouse button or by pressing SHIFT+F10 or the Applications key.

This event is especially useful when the control does not have an associated popup menu (the PopupMenu property is not set) or if the AutoPopup property of the control's associated popup menu is false. However, the OnContextPopup can also be used to override the automatic context menu that appears when the control has an associated popup menu with an AutoPopup property of true. In this last case, if the event handler displays its own menu, it should set the Handled parameter to true to suppress the default context menu.

The handler's MousePos parameter indicates the position of the mouse, in client coordinates.. If the event was not generated by a mouse click, MousePos is (-1,-1).

Note: Parent controls receive an OnContextPopup event before their child controls. In addition, for many child controls, the default window procedure causes the parent control to receive an OnContextPopup event after the child control. As a result, when parent controls do not set Handled to true in an OnContextPopup event handler, the event handler may be called multiple times for each context menu invocation.

See Also

TPopupMenu (see page 1430)

PopupMenu

AutoPopup (see page 1437)

39.1.2.6.3 TMediaPlayer.OnEnter Event

Occurs when a control receives the input focus.

Pascal

```
property OnEnter;
```

C++

```
__property OnEnter;
```

Description

Use the OnEnter event handler to cause any special processing to occur when a control becomes active.

The OnEnter event does not occur when switching between forms or between another application and the application that includes the control.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnEnter event occurs for the container before the OnEnter event of the contained control.

Similarly, an OnExit event of the container occurs after the OnExit event of the control in a container when focus moves to another control outside the container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

C++ Examples:

```
/*
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
*/
TWinControl *myActiveControl;

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  FontDialog1->Options << fdApplyButton;
  FontDialog1->Execute();
}

void __fastcall TForm1::FontDialog1Apply(TObject *Sender, HWND Wnd)
{
  if (myActiveControl->ClassNameIs("TEdit"))
   (dynamic_cast<TEdit *>(myActiveControl))->Font->Assign(FontDialog1->Font);
  else if (myActiveControl->ClassNameIs("TRichEdit"))
    (dynamic_cast<TRichEdit *>(myActiveControl))->SelAttributes->Assign(FontDialog1->Font);
  else
    MessageBeep(0);
}

void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  myActiveControl = ActiveControl;
```

```
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
   FontDialog1.Options := FontDialog1.Options + [fdApplyButton];
   FontDialog1.Execute;
end;

procedure TForm1.Edit1Enter(Sender: TObject);
begin
  myActiveControl := ActiveControl;
end;

procedure TForm1.FontDialog1Apply(Sender: TObject; Wnd: HWND);
begin
  if myActiveControl is TEdit then
    with myActiveControl as TEdit do
       Font.Assign(TFontDialog(Sender).Font)
  else if myActiveControl is TRichEdit then
    with myActiveControl as TRichEdit do
      SelAttributes.Assign(TFontDialog(Sender).Font)
  else
    Beep;
end;
{
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
```

```
  inactive, the color of the control returns to the Windows
  system color for a window.  These event handlers could also
  be shared.
  }
  procedure TForm1.Edit1Enter(Sender: TObject);
  begin
    Edit1.Color := clYellow;
  end;

  procedure TForm1.Edit1Exit(Sender: TObject);
  begin
    Edit1.Color := clWindow;
  end;

  procedure TForm1.Memo1Enter(Sender: TObject);
  begin
    Memo1.Color := clYellow;
  end;

  procedure TForm1.Memo1Exit(Sender: TObject);
  begin
    Memo1.Color := clWindow;
  end;
```

See Also

OnExit

ActiveControl

OnActivate

Using a Single Navigator for Multiple Datasets

39.1.2.6.4 TMediaPlayer.OnExit Event

Occurs when the input focus shifts away from one control to another.

Pascal

```
property OnExit;
```

C++

```
__property OnExit;
```

Description

Use the OnExit event handler to provide special processing when the control ceases to be active.

The OnExit event does not occur when switching between forms or between another application and your application.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnExit event occurs for the control inside the container before the OnExit event of the container.

Similarly, an OnEnter event of the container occurs before the OnEnter event of the control in a container when focus moves to a control inside a container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

Note: In some control classes the ActiveControl property updates before the OnExit event occurs.

C++ Examples:

```
/*
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255 ,
as this validation test cannot be accomplished
using the edit mask property alone.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  // initialize the maskedit boxes to contain the empty string
  MaskEdit1->Text = "";
  MaskEdit2->Text = "";
  MaskEdit3->Text = "";

  // use the Courier font in the maskedit boxes
  MaskEdit1->Font->Name = "Courier";
  MaskEdit2->Font->Name = "Courier";
  MaskEdit3->Font->Name = "Courier";

  // initialize the mask of each maskedit box
  // according to IP address coding conventions
  MaskEdit1->EditMask = "!099.099.099.099;1; ";
  MaskEdit2->EditMask = "!099.099.099.099;1; ";
  MaskEdit3->EditMask = "!099.099.099.099;1; ";

  MaskEdit2->OnExit = MaskEdit2Exit;
}

void __fastcall TForm1::MaskEdit2Exit(TObject *Sender)
{
  // extract the net and host address from the IP
  String IP = MaskEdit1->Text;
  int net1 = IP.SubString(0, 3).TrimRight().ToInt();
  int net2 = IP.SubString(5, 3).TrimRight().ToInt();
  int host1 = IP.SubString(9, 3).TrimRight().ToInt();
  int host2 = IP.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (net1 < 0 || net1 > 255 ||
      net2 < 0 || net2 > 255 ||
      host1 < 0 || host1 > 255 ||
      host2 < 0 || host2 > 255)
    throw(Exception("Not a valid IP address."));

  // extract the net and host mask from the subnet mask
  String mask = MaskEdit2->Text;
  int netmask1 = mask.SubString(0, 3).TrimRight().ToInt();
  int netmask2 = mask.SubString(5, 3).TrimRight().ToInt();
  int hostmask1 = mask.SubString(9, 3).TrimRight().ToInt();
  int hostmask2 = mask.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (netmask1 < 0 || netmask1 > 255 ||
      netmask2 < 0 || netmask2 > 255 ||
```

```
        hostmask1 < 0 || hostmask1 > 255 ||
        hostmask2 < 0 || hostmask2 > 255)
      throw(Exception("Not a valid subnet mask."));

    // compute the subnet address using bit-wise AND
    int sub_net1 = net1 & netmask1;
    int sub_net2 = net2 & netmask2;
    int sub_host1 = host1 & hostmask1;
    int sub_host2 = host2 & hostmask2;

    // display the subnet address
    MaskEdit3->Text =
      String(sub_net1) + "." + String(sub_net2) + "." +
      String(sub_host1) + "." + String(sub_host2);
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255,
as this validation test cannot be accomplished
using the edit mask property alone.
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  { initialize the maskedit boxes to contain the empty string }
  MaskEdit1.Text := '';
  MaskEdit2.Text := '';
  MaskEdit3.Text := '';
```

```
  { use the Courier font in the maskedit boxes }
  MaskEdit1.Font.Name := 'Courier';
  MaskEdit2.Font.Name := 'Courier';
  MaskEdit3.Font.Name := 'Courier';

  {
  initialize the mask of each maskedit box
  according to IP address coding conventions
  }
  MaskEdit1.EditMask := '!099.099.099.099;1; ';
  MaskEdit2.EditMask := '!099.099.099.099;1; ';
  MaskEdit3.EditMask := '!099.099.099.099;1; ';

  MaskEdit2.OnExit := MaskEdit2Exit;
end;

procedure TForm1.MaskEdit2Exit(Sender: TObject);
var
  net1, net2, host1, host2,
  netmask1, netmask2, hostmask1, hostmask2,
  sub_net1, sub_net2, sub_host1, sub_host2: Integer;

  IP, mask: String;

begin
  // extract the net and host address from the IP
  IP := MaskEdit1.Text;
  net1 := StrToInt(TrimRight(Copy(IP, 0, 3)));
  net2 := StrToInt(TrimRight(Copy(IP, 5, 3)));
  host1 := StrToInt(TrimRight(Copy(IP, 9, 3)));
  host2 := StrToInt(TrimRight(Copy(IP, 13, 3)));

  // a range test that you cannot validate through edit masks
  if ((net1 < 0) Or (net1 > 255) Or
      (net2 < 0) Or (net2 > 255) Or
      (host1 < 0) Or (host1 > 255) Or
      (host2 < 0) Or (host2 > 255)) then
    raise EArgumentException.Create('Not a valid IP address.');

  // extract the net and host mask from the subnet mask
  mask := MaskEdit2.Text;
  netmask1 := StrToInt(TrimRight(Copy(mask, 0, 3)));
  netmask2 := StrToInt(TrimRight(Copy(mask, 5, 3)));
  hostmask1 := StrToInt(TrimRight(Copy(mask, 9, 3)));
  hostmask2 := StrToInt(TrimRight(Copy(mask, 13, 3)));

  // a range test that you cannot validate through edit masks
  if ((netmask1 < 0) Or (netmask1 > 255) Or
      (netmask2 < 0) Or (netmask2 > 255) Or
      (hostmask1 < 0) Or (hostmask1 > 255) Or
      (hostmask2 < 0) Or (hostmask2 > 255)) then
    raise EArgumentException.Create('Not a valid subnet mask.');

  // compute the subnet address using bit-wise AND
  sub_net1 := net1 And netmask1;
  sub_net2 := net2 And netmask2;
  sub_host1 := host1 And hostmask1;
  sub_host2 := host2 And hostmask2;

  // display the subnet address
  MaskEdit3.Text :=
    IntToStr(sub_net1) + '.' + IntToStr(sub_net2) + '.' +
    IntToStr(sub_host1) + '.' + IntToStr(sub_host2);
end;
{
This example uses an edit box and a memo control on a form.
```

```
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
}
procedure TForm1.Edit1Enter(Sender: TObject);
begin
  Edit1.Color := clYellow;
end;

procedure TForm1.Edit1Exit(Sender: TObject);
begin
  Edit1.Color := clWindow;
end;

procedure TForm1.Memo1Enter(Sender: TObject);
begin
  Memo1.Color := clYellow;
end;

procedure TForm1.Memo1Exit(Sender: TObject);
begin
  Memo1.Color := clWindow;
end;
```

See Also

OnEnter

ActiveControl

39.1.2.6.5 TMediaPlayer.OnMouseEnter Event

Occurs when the user moves the mouse into a control.

Pascal

```
property OnMouseEnter;
```

C++

```
__property OnMouseEnter;
```

Description

Use the OnMouseEnter event handler to implement any special processing that should occur as a result of moving the mouse into a control when the parent top-level form is not active.

39.1.2.6.6 TMediaPlayer.OnMouseLeave Event

Occurs when the user moves the mouse outside of a control.

Pascal

```
property OnMouseLeave;
```

C++

```
__property OnMouseLeave;
```

Description

Use the OnMouseLeave event handler to implement any special processing that should occur as a result of moving the mouse outside of a control when the parent top-level form is not active.

39.1.2.6.7 TMediaPlayer.OnNotify Event

Occurs upon the completion of a media control method.

Pascal

```
property OnNotify: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnNotify;
```

Description

OnNotify occurs upon the completion of a media control method (Back (see page 2313), Close (see page 2314), Eject (see page 2315), Next (see page 2318), Open (see page 2319), Pause (see page 2322), PauseOnly (see page 2323), Play (see page 2323), Previous (see page 2327), Resume (see page 2327), Rewind (see page 2328), StartRecording (see page 2329), Step (see page 2330), or Stop (see page 2330)) when the Notify (see page 2350) property is set to true before the call to the media control method. After an OnNotify event, the Notify (see page 2350) property must be reset to true for the next OnNotify event to occur.

See Also

DoNotify (see page 2315)

MMNotify (see page 2318)

Notify (see page 2350)

39.1.2.6.8 TMediaPlayer.OnPostClick Event

Occurs after the OnClick (see page 2366) event handler is called.

Pascal

```
property OnPostClick: EMPPostNotify;
```

C++

```
__property EMPPostNotify OnPostClick;
```

Description

OnPostClick is generated after the code of the OnClick (see page 2366) event handler has been called. If Wait (see page 2365) is true when the media player was clicked, OnPostClick won't be called until the completion of the OnClick (see page 2366) code. If Wait (see page 2365) is false, control can return to the application before completion of the OnClick (see page 2366) code; therefore, the OnPostClick event may occur before the actions initiated by the OnClick (see page 2366) event have completed.

For example, if the user clicks the Play (see page 2323) button and the DoDefault parameter of the OnClick (see page 2366) event handler for the media player is true, the media is played. If the media is long enough, it will still be playing when the OnPostClick event is generated if Wait (see page 2365) is true. If Wait (see page 2365) is false, however, OnPostClick won't occur until the media has finished playing.

See Also

OnClick (see page 2366)

PostClick (see page 2326)

Wait (see page 2365)

39.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TMPBtnType (see page 2376)	TMPBtnType defines the possible buttons in a TMediaPlayer (see page 2305) component and TButtonSet (see page 2385) represents a set of those buttons.
	TMPDevCaps (see page 2378)	TMPDevCaps defines the capabilities of a TMediaPlayer (see page 2305) component and TMPDevCapsSet (see page 2385) represents a set of those capabilities.
	TMPDeviceTypes (see page 2378)	TMPDeviceTypes identifies a multimedia device.
	TMPGlyph (see page 2380)	This is record MPlayer.TMPGlyph.
	TMPModes (see page 2380)	TMPModes identifies the state of a TMediaPlayer (see page 2305) component.
	TMPNotifyValues (see page 2381)	TMPNotifyValues identifies the types of notifications a media device can give a TMediaPlayer (see page 2305) component.
	TMPTimeFormats (see page 2382)	TMPTimeFormats indicates the format a media player uses to indicate position.

Records

	Name	Description
	TMPButton (see page 2377)	This is record MPlayer.TMPButton.

39.2.1 MPlayer.TMPBtnType Enumeration

TMPBtnType defines the possible buttons in a TMediaPlayer (see page 2305) component and TButtonSet (see page 2385) represents a set of those buttons.

Pascal

```
TMPBtnType = (
  btPlay,
  btPause,
  btStop,
  btNext,
  btPrev,
  btStep,
  btBack,
  btRecord,
  btEject
);
```

C++

```
enum TMPBtnType {
  btPlay,
  btPause,
  btStop,
  btNext,
  btPrev,
  btStep,
  btBack,
  btRecord,
  btEject
};
```

File

MPlayer

Description

TMPBtnType represents the type of a media player button. TButtonSet (see page 2385) is a set of button types for the media player.

Each button type has an associated action on the media player. The following table lists the button types and their associated actions:

Value	Action
btPlay	Plays the media player.
btRecord	Starts recording.
btStop	Stops playing or recording.
btNext	Skips to the next track, or to the end if the medium doesn't use tracks.
btPrev	Skips to the previous track, or to the beginning if the medium doesn't use tracks.
btStep	Moves forward a number of frames.
btBack	Moves backward a number of frames.
btPause	Pauses playing or recording. If already paused when clicked, resumes playing or recording.
btEject	Ejects the medium.

See Also

TBitBtn

TSpeedButton

39.2.2 MPlayer.TMPButton Record

Pascal

```
TMPButton = record
  Visible: Boolean;
  Enabled: Boolean;
  Colored: Boolean;
  Auto: Boolean;
  Bitmaps: array[TMPGlyph] of TBitmap;
end;
```

C++

```
struct TMPButton {
  Boolean Visible;
  Boolean Enabled;
  Boolean Colored;
  Boolean Auto;
  array[TMPGlyph] of TBitmap Bitmaps;
};
```

File

MPlayer

Description

This is record MPlayer.TMPButton.

39.2.3 MPlayer.TMPDevCaps Enumeration

TMPDevCaps defines the capabilities of a TMediaPlayer (see page 2305) component and TMPDevCapsSet (see page 2385) represents a set of those capabilities.

Pascal

```
TMPDevCaps = (
  mpCanStep,
  mpCanEject,
  mpCanPlay,
  mpCanRecord,
  mpUsesWindow
);
```

C++

```
enum TMPDevCaps {
  mpCanStep,
  mpCanEject,
  mpCanPlay,
  mpCanRecord,
  mpUsesWindow
};
```

File

MPlayer

Description

TMPDevCaps represents the capabilities of a media player control. TMPDevCapsSet (see page 2385) is a set of TMPDevCaps values.

The following table lists the possible TMPDevCaps values:

Value	Capability
mpCanEject	Can eject media
mpCanPlay	Can play media
mpCanRecord	Can record media
mpCanStep	Can step forward or backward within media
mpUsesWindow	Uses a window for displaying output

See Also

TMediaPlayer (see page 2305)

39.2.4 MPlayer.TMPDeviceTypes Enumeration

TMPDeviceTypes identifies a multimedia device.

Pascal

```
TMPDeviceTypes = (
  dtAutoSelect,
  dtAVIVideo,
  dtCDAudio,
  dtDAT,
  dtDigitalVideo,
  dtMMMovie,
  dtOther,
  dtOverlay,
  dtScanner,
  dtSequencer,
  dtVCR,
  dtVideodisc,
  dtWaveAudio
);
```

C++

```
enum TMPDeviceTypes {
  dtAutoSelect,
  dtAVIVideo,
  dtCDAudio,
  dtDAT,
  dtDigitalVideo,
  dtMMMovie,
  dtOther,
  dtOverlay,
  dtScanner,
  dtSequencer,
  dtVCR,
  dtVideodisc,
  dtWaveAudio
};
```

File

MPlayer

Description

TMPDeviceTypes type identifies the multimedia device types that can be opened by a TMediaPlayer (see page 2305) component.

The following table lists the possible values:

Value	Meaning
dtAutoSelect	the device type is determined by the file extension.
dtAVIVideo	AVI clip
dtCDAudio	Audio CD recording
dtDAT	DAT file
dtDigitalVideo	Digital video recording
dtMMMovie	MM recording
dtOther	media type not identified by other categories
dtOverlay	Overlay
dtScanner	scanner
dtSequencer	sequencer
dtVCR	VCR

dtVideodisc	Videodisc
dtWaveAudio	.wav file

See Also

TMediaPlayer (see page 2305)

39.2.5 MPlayer.TMPGlyph Enumeration

Pascal

```
TMPGlyph = (
  mgEnabled,
  mgDisabled,
  mgColored
);
```

C++

```
enum TMPGlyph {
  mgEnabled,
  mgDisabled,
  mgColored
};
```

File

MPlayer

Description

This is record MPlayer.TMPGlyph.

39.2.6 MPlayer.TMPModes Enumeration

TMPModes identifies the state of a TMediaPlayer (see page 2305) component.

Pascal

```
TMPModes = (
  mpNotReady,
  mpStopped,
  mpPlaying,
  mpRecording,
  mpSeeking,
  mpPaused,
  mpOpen
);
```

C++

```
enum TMPModes {
  mpNotReady,
  mpStopped,
  mpPlaying,
  mpRecording,
  mpSeeking,
  mpPaused,
  mpOpen
```

```
};
```

File

MPlayer

Description

TMPModes type identifies the state of a media player. The following table lists the possible values:

Value	Mode
mpNotReady	Not ready
mpStopped	Stopped
mpPlaying	Playing
mpRecording	Recording
mpSeeking	Seeking
mpPaused	Paused
mpOpen	Open

See Also

TMediaPlayer (see page 2305)

39.2.7 MPlayer.TMPNotifyValues Enumeration

TMPNotifyValues identifies the types of notifications a media device can give a TMediaPlayer (see page 2305) component.

Pascal

```
TMPNotifyValues = (
  nvSuccessful,
  nvSuperseded,
  nvAborted,
  nvFailure
);
```

C++

```
enum TMPNotifyValues {
  nvSuccessful,
  nvSuperseded,
  nvAborted,
  nvFailure
};
```

File

MPlayer

Description

The TMPNotifyValues type defines the notification values for a multimedia device used with a TMediaPlayer (see page 2305). The possible values for the TMPNotifyValues type are as follows.

Value	Meaning
nvSuccessful	Command completed successfully
nvSuperseded	Command was superseded by another command

nvAborted	Command was aborted by the user
nvFailure	Command failed

See Also

TMediaPlayer (see page 2305)

39.2.8 MPlayer.TMPTimeFormats Enumeration

TMPTimeFormats indicates the format a media player uses to indicate position.

Pascal

```
TMPTimeFormats = (
  tfMilliseconds,
  tfHMS,
  tfMSF,
  tfFrames,
  tfSMPTE24,
  tfSMPTE25,
  tfSMPTE30,
  tfSMPTE30Drop,
  tfBytes,
  tfSamples,
  tfTMSF
);
```

C++

```
enum TMPTimeFormats {
  tfMilliseconds,
  tfHMS,
  tfMSF,
  tfFrames,
  tfSMPTE24,
  tfSMPTE25,
  tfSMPTE30,
  tfSMPTE30Drop,
  tfBytes,
  tfSamples,
  tfTMSF
};
```

File

MPlayer

Description

The TMPTimeFormats type defines the time formats for a multimedia device used with a TMediaPlayer (see page 2305). The following table lists the possible values for the TMPTimeFormats type:

Value	Time format
tfMilliseconds	Milliseconds are stored as a 4-byte integer variable.
tfHMS	Hours, minutes, and seconds packed into a 4-byte integer. From least significant to most significant byte, the data values are
Hours (least significant byte)	
Minutes	

Seconds	
Unused (most significant byte)	
tfMSF	Minutes, seconds, and frames packed into a 4-byte integer. From least significant to most significant byte, the data values are
Minutes (least significant byte)	
Seconds	
Frames	
Unused (most significant byte)	
tfFrames	Frames are stored as a 4-byte integer variable.
tfSMPTE24	24-frame SMPTE packs values in a 4-byte variable. From least significant to most significant byte, the data values are
Hours (least significant byte)	
Minutes	
Seconds	
Frames (most significant byte)	
SMPTE (Society of Motion Picture and Television Engineers) time is an absolute time format expressed in hours, minutes, seconds, and frames. The standard SMPTE division types are 24, 25, and 30 frames per second.	
tfSMPTE25	25-frame SMPTE packs data into a 4-byte variable in the same order as 24-frame SMPTE.
tfSMPTE30	30-frame SMPTE packs data into the 4-byte variable in the same order as 24-frame SMPTE.
tfSMPTE30Drop	30-drop-frame SMPTE packs data into the 4-byte variable in the same order as 24-frame SMPTE.
tfBytes	Bytes are stored as a 4-byte integer variable.
tfSamples	Samples are stored as a 4-byte integer variable.
tfTMSF	Tracks, minutes, seconds, and frames are packed in the 4-byte variable. From least significant to most significant byte, the data values are
Tracks (least significant byte)	
Minutes	
Seconds	
Frames (most significant byte)	

Note that MCI uses continuous track numbering.

Note: Functions provided with MCI to help decoding the 4-byte integer specified in a given time format are documented under MCI Macros for Encoding and Decoding Time Data in the MMSYSTEM.HLP Help file.

See Also

TMediaPlayer (see page 2305)

39.3 Types

The following table lists types in this documentation.

Types

Name	Description
EMPNotify (see page 2384)	EMPNotify is the type of event handlers that respond when the user interacts with a TMediaPlayer (see page 2305) component.
EMPPostNotify (see page 2384)	EMPPostNotify is the type of a media player's OnPostClick event handler.
TButtonSet (see page 2385)	TButtonSet is a set of TMPBtnType (see page 2376) values.
TMediaDevice (see page 2385)	This is type MPlayer.TMediaDevice.
TMPDevCapsSet (see page 2385)	This is type MPlayer.TMPDevCapsSet.

39.3.1 MPlayer.EMPNotify Type

EMPNotify is the type of event handlers that respond when the user interacts with a TMediaPlayer (see page 2305) component.

Pascal

```
EMPNotify = procedure (Sender: TObject; Button: TMPBtnType; var DoDefault: Boolean) of object;
```

C++

```
(Sender: TObject; Button: TMPBtnType; var DoDefault: Boolean) ( EMPNotify)();
```

File

MPlayer

Description

Sender is the TMediaPlayer (see page 2305) component whose button was clicked.

Button indicates the button that was clicked. It can be one of the following values: btBack, btEject, btNext, btPause, btPlay, btPrev, btRecord, btStep, or btStop.

DoDefault returns an indication of whether the media player should make its default response after the event handler exits.

See Also

TMediaPlayer (see page 2305)

39.3.2 MPlayer.EMPPostNotify Type

EMPPostNotify is the type of a media player's OnPostClick event handler.

Pascal

```
EMPPostNotify = procedure (Sender: TObject; Button: TMPBtnType) of object;
```

C++

```
(Sender: TObject; Button: TMPBtnType) ( EMPPostNotify)();
```

File

MPlayer

Description

Sender is the TMediaPlayer (see page 2305) component whose button was clicked.

Button indicates the button that was clicked. It can be one of the following values: btBack, btEject, btNext, btPause, btPlay, btPrev, btRecord, btStep, or btStop.

See Also

TMediaPlayer (see page 2305)

39.3.3 MPlayer.TButtonSet Type

TButtonSet is a set of TMPBtnType (see page 2376) values.

Pascal

```
TButtonSet = set of TMPBtnType;
```

C++

```
TMPBtnType TButtonSet;
```

File

MPlayer

Description

TButtonSet represents a set of zero or more of the buttons that can appear on a TMPDeviceType control.

39.3.4 MPlayer.TMediaDevice Type

Pascal

```
TMediaDevice = Word;
```

C++

```
Word TMediaDevice;
```

File

MPlayer

Description

This is type MPlayer.TMediaDevice.

39.3.5 MPlayer.TMPDevCapsSet Type

Pascal

```
TMPDevCapsSet = set of TMPDevCaps;
```

C++

```
TMPDevCaps TMPDevCapsSet;
```

File

MPlayer

Description

This is type MPlayer.TMPDevCapsSet.

40 Mtsobj Namespace

This is namespace Mtsobj.

Topics

Name	Description
Classes (see page 2388)	The following table lists classes in this documentation.

Classes

	Name	Description
	TMtsAutoObject (see page 2388)	TMtsAutoObject encapsulates the objects and interfaces of the MTS application server.

40.1 Classes

The following table lists classes in this documentation.

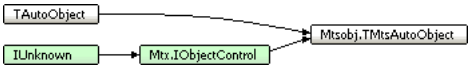
Classes

	Name	Description
	TMtsAutoObject (see page 2388)	TMtsAutoObject encapsulates the objects and interfaces of the MTS application server.

40.1.1 TMtsAutoObject Class

TMtsAutoObject encapsulates the objects and interfaces of the MTS application server.

Class Hierarchy



Pascal

```
TMtsAutoObject = class(TAutoObject, IObjectControl);
```

C++

```
class TMtsAutoObject : public TAutoObject, public IObjectControl;
```

File

Mtsobj

Description

Use a TMtsAutoObject object as the central repository for all objects in an MTS-enabled application server.

MTS Automation objects are dual-interface Automation servers that implement the IObjectControl (see page 2424) interface. MTS Automation objects are like any Automation server with additional features that encapsulate MTS services such as transaction support and security.

See Also

TMtsDataModule (see page 2398)

Writing Applications Using COM

40.1.1.1 TMtsAutoObject Members

The following tables list the members exposed by TMtsAutoObject.

Protected Methods

	Name	Description
	Activate (see page 2389)	This is Activate, a member of class TMtsAutoObject.
	CanBePooled (see page 2390)	This is CanBePooled, a member of class TMtsAutoObject.
	Deactivate (see page 2390)	This is Deactivate, a member of class TMtsAutoObject.

	OnActivate (see page 2393)	This is OnActivate, a member of class TMtsAutoObject.
	OnDeactivate (see page 2393)	This is OnDeactivate, a member of class TMtsAutoObject.

Protected Properties

	Name	Description
	ObjectContext (see page 2394)	This is ObjectContext, a member of class TMtsAutoObject.

Public Methods

	Name	Description
	DisableCommit (see page 2390)	Prevents clients from completing transactions.
	EnableCommit (see page 2391)	Allows clients to complete transactions without releasing state information.
	IsCallerInRole (see page 2391)	Indicates whether the current client matches a particular security role.
	IsInTransaction (see page 2392)	Indicates whether the MTS Automation object is running in a transaction.
	IsSecurityEnabled (see page 2392)	Indicates whether security is enabled for the current MTS Automation object instance.
	SetAbort (see page 2393)	Allows the MTS Automation object to be deactivated and aborts any current transactions.
	SetComplete (see page 2393)	Indicates that the MTS Automation object has successfully completed its current task.

Public Properties

	Name	Description
	Pooled (see page 2394)	Indicates whether the object can be pooled when installed under COM+.

40.1.1.2 TMtsAutoObject Methods

The methods of the TMtsAutoObject class are listed here.

Protected Methods

	Name	Description
	Activate (see page 2389)	This is Activate, a member of class TMtsAutoObject.
	CanBePooled (see page 2390)	This is CanBePooled, a member of class TMtsAutoObject.
	Deactivate (see page 2390)	This is Deactivate, a member of class TMtsAutoObject.
	OnActivate (see page 2393)	This is OnActivate, a member of class TMtsAutoObject.
	OnDeactivate (see page 2393)	This is OnDeactivate, a member of class TMtsAutoObject.

Public Methods

	Name	Description
	DisableCommit (see page 2390)	Prevents clients from completing transactions.
	EnableCommit (see page 2391)	Allows clients to complete transactions without releasing state information.
	IsCallerInRole (see page 2391)	Indicates whether the current client matches a particular security role.
	IsInTransaction (see page 2392)	Indicates whether the MTS Automation object is running in a transaction.
	IsSecurityEnabled (see page 2392)	Indicates whether security is enabled for the current MTS Automation object instance.
	SetAbort (see page 2393)	Allows the MTS Automation object to be deactivated and aborts any current transactions.
	SetComplete (see page 2393)	Indicates that the MTS Automation object has successfully completed its current task.

40.1.1.2.1 TMtsAutoObject.Activate Method

Pascal

```
procedure Activate; safecall;
```

C++

```
__fastcall Activate();
```

Description

This is Activate, a member of class TMtsAutoObject.

40.1.1.2.2 TMtsAutoObject.CanBePooled Method

Pascal

```
function CanBePooled: Bool; virtual; stdcall;
```

C++

```
virtual __fastcall __stdcall Bool CanBePooled();
```

Description

This is CanBePooled, a member of class TMtsAutoObject.

40.1.1.2.3 TMtsAutoObject.Deactivate Method

Pascal

```
procedure Deactivate; stdcall;
```

C++

```
__fastcall __stdcall Deactivate();
```

Description

This is Deactivate, a member of class TMtsAutoObject.

40.1.1.2.4 TMtsAutoObject.DisableCommit Method

Prevents clients from completing transactions.

Pascal

```
procedure DisableCommit;
```

C++

```
__fastcall DisableCommit();
```

Description

Call DisableCommit to indicate that the MTS Automation object cannot complete its current transaction or release state information until it receives further method invocations from the client. After calling DisableCommit, the current transaction (if any) cannot be completed until the MTS Automation object calls EnableCommit (see page 2391) or SetComplete (see page 2393).

With transactions that are started automatically when the MTS Automation object's interface is called, DisableCommit is not necessary. These transactions are not committed until the MTS Automation object calls SetComplete (see page 2393). However, when the MTS Automation object is enlisted in a transaction that was started previously, the client controls the transaction and can attempt to commit it before the MTS Automation object has completed its part. Calling DisableCommit prevents the client from prematurely committing a transaction. If a client tries to commit a transaction after the MTS Automation object has called DisableCommit, the transaction aborts.

Tip: Call DisableCommit at the beginning of a method to protect your MTS Automation object from committing prematurely in the event of an unexpected return. If the method call finishes normally, calling EnableCommit (see page 2391) or SetComplete

(see page 2393) reverses this affect and allows the transaction to complete.

See Also

EnableCommit (see page 2391)

SetAbort (see page 2393)

SetComplete (see page 2393)

IsInTransaction (see page 2392)

40.1.1.2.5 TMtsAutoObject.EnableCommit Method

Allows clients to complete transactions without releasing state information.

Pascal

```
procedure EnableCommit;
```

C++

```
__fastcall EnableCommit();
```

Description

Call EnableCommit to indicate that the MTS Automation object can complete its current transaction without releasing state information. Unlike the SetComplete (see page 2393) method (which also signals that the current transaction can be committed), EnableCommit does not allow MTS to deactivate the MTS Automation object. This lets the MTS Automation object preserve state information over multiple interface calls, while still allowing clients that have enlisted the object in a transaction to commit.

By default, transactions can be completed when an interface call ends, unless the DisableCommit (see page 2390) method is called. There is no need to call EnableCommit unless the MTS Automation object has previously called DisableCommit (see page 2390).

Note: Although EnableCommit permits the current transaction to be completed, it does not automatically commit the transaction. The client that enlisted the MTS Automation object must still explicitly commit that transaction. If MTS started the transaction automatically (because the MTS Automation object requires a transaction for its interface calls), the transaction is not committed until the data module calls SetComplete (see page 2393).

See Also

DisableCommit (see page 2390)

SetComplete (see page 2393)

SetAbort (see page 2393)

IsInTransaction (see page 2392)

40.1.1.2.6 TMtsAutoObject.IsCallerInRole Method

Indicates whether the current client matches a particular security role.

Pascal

```
function IsCallerInRole(const Role: WideString): Bool;
```

C++

```
__fastcall Bool IsCallerInRole(const BSTR Role);
```

Description

MTS provides role-based security. Each level of access to your application server can be assigned a role. Using the MTS Explorer, administrators can prevent clients from using an interface based on their role. (This is called declarative security.) In addition, however, programs can provide more fine-grained security programmatically. To do this, use IsCallerInRole to provide access to blocks of code based on the caller's security role.

IsCallerInRole checks whether the current client is part of the role named by the Role parameter. It returns true if the client matches the named role, and False otherwise. Use IsCallerInRole to prevent clients from invoking certain blocks of code if they do not have the appropriate security role.

Note: MTS security is not available on servers running Windows95. If the MTS application is running on Windows95, IsCallerInRole always returns true . In addition, IsCallerInRole always returns True if the application is part of an MTS Library package (which runs in the client's process space).

40.1.1.2.7 TMtsAutoObject.IsInTransaction Method

Indicates whether the MTS Automation object is running in a transaction.

Pascal

```
function IsInTransaction: Bool;
```

C++

```
__fastcall Bool IsInTransaction();
```

Description

Call IsInTransaction to determine whether the MTS Automation object is currently running in a transaction. For example, if the MTS Automation object requires a transaction, use IsInTransaction to check that this requirement has not been turned off by an administrator using the MTS Explorer, and to raise an exception if it has.

Note: If IsInTransaction returns False, the EnableCommit (see page 2391) and DisableCommit (see page 2390) methods have no effect.

See Also

EnableCommit (see page 2391)

DisableCommit (see page 2390)

40.1.1.2.8 TMtsAutoObject.IsSecurityEnabled Method

Indicates whether security is enabled for the current MTS Automation object instance.

Pascal

```
function IsSecurityEnabled: Bool;
```

C++

```
__fastcall Bool IsSecurityEnabled();
```

Description

Call IsSecurityEnabled to determine whether the MTS Automation object is currently protected by MTS security services. MTS security is enabled unless the object is running in the client's process. This can happen if the MTS Automation object is configured to run in the client's process space by an administrator using the MTS Explorer, or if the client application is in the same MTS package as the MTS Automation object.

Note: MTS security is not available for servers running on Windows95.

40.1.1.2.9 TMtsAutoObject.OnActivate Method

Pascal

```
procedure OnActivate; virtual;
```

C++

```
virtual __fastcall OnActivate();
```

Description

This is OnActivate, a member of class TMtsAutoObject.

40.1.1.2.10 TMtsAutoObject.OnDeactivate Method

Pascal

```
procedure OnDeactivate; virtual;
```

C++

```
virtual __fastcall OnDeactivate();
```

Description

This is OnDeactivate, a member of class TMtsAutoObject.

40.1.1.2.11 TMtsAutoObject.SetAbort Method

Allows the MTS Automation object to be deactivated and aborts any current transactions.

Pascal

```
procedure SetAbort;
```

C++

```
__fastcall SetAbort();
```

Description

Call SetAbort to indicate that any current transaction that includes the MTS Automation object to abort. If MTS started a transaction automatically, the transaction aborts when the current method call exits. If the MTS Automation object was enlisted in a pre-existing transaction, the client transaction will eventually fail. Usually, SetAbort is called in the except block of a try...except statement.

In addition to aborting any current transaction, SetAbort indicates that the MTS Automation object can be deactivated when the current interface call exits. This means that state information is not preserved for the next time and the client calls the Automation object's interface. To prevent transactions from committing successfully without losing state information, use the DisableCommit (see page 2390) method instead.

Tip: Call SetAbort at the beginning of a method to protect your MTS data module from committing prematurely in the event of an unexpected return. If the method call finishes normally, calling SetComplete (see page 2393) reverses this affect and allows the transaction to complete.

40.1.1.2.12 TMtsAutoObject.SetComplete Method

Indicates that the MTS Automation object has successfully completed its current task.

Pascal

```
procedure SetComplete;
```

C++

```
__fastcall SetComplete();
```

Description

Call SetComplete to indicate that the MTS Automation object no longer needs to maintain state information. After calling SetComplete, the MTS Automation object is deactivated when the current method call exits.

If the MTS Automation object is part of a transaction, SetComplete indicates that this part of the transaction can commit. If MTS started the transaction automatically when the client called the MTS Automation object's methods, MTS tries to commit the transaction when the current method exits.

Note: This transaction may still fail if the current transaction included another MTS object that called SetAbort (see page 2393) or DisableCommit (see page 2390).

See Also

EnableCommit (see page 2407)

40.1.1.3 TMtsAutoObject Properties

The properties of the TMtsAutoObject class are listed here.

Protected Properties

	Name	Description
	ObjectContext (see page 2394)	This is ObjectContext, a member of class TMtsAutoObject.

Public Properties

	Name	Description
	Pooled (see page 2394)	Indicates whether the object can be pooled when installed under COM+.

40.1.1.3.1 TMtsAutoObject.ObjectContext Property

Pascal

```
property ObjectContext: IObjectContext;
```

C++

```
__property IObjectContext ObjectContext;
```

Description

This is ObjectContext, a member of class TMtsAutoObject.

40.1.1.3.2 TMtsAutoObject.Pooled Property

Indicates whether the object can be pooled when installed under COM+.

Pascal

```
property Pooled: Boolean;
```

C++

```
__property Boolean Pooled;
```

Description

Set Pooled to indicate whether instances of the MTS Automation object can be pooled when installed under COM+. The MTS Automation object uses the Pooled property when implementing the IObjectControl (see page 2424) method CanBePooled (see page 2390). This instructs COM+ that the object can be moved to an object pool when deactivated, so that the single instance is shared by all clients.

Warning: Do not set Pooled to True unless the MTS Automation object is stateless.

See Also

Pooling Remote Data Modules

41 MtsRdm Namespace

This is namespace MtsRdm.

Topics

Name	Description
Classes (see page 2398)	The following table lists classes in this documentation.

Classes

	Name	Description
	TMtsDataModule (see page 2398)	TMtsDataModule encapsulates the objects and interfaces of a transactional application server.

41.1 Classes

The following table lists classes in this documentation.

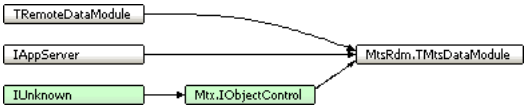
Classes

	Name	Description
	TMtsDataModule (see page 2398)	TMtsDataModule encapsulates the objects and interfaces of a transactional application server.

41.1.1 TMtsDataModule Class

TMtsDataModule encapsulates the objects and interfaces of a transactional application server.

Class Hierarchy



Pascal

```
TMtsDataModule = class(TRemoteDataModule, IAppServer, IObjectControl);
```

C++

```
class TMtsDataModule : public TRemoteDataModule, public IAppServer, public IObjectControl;
```

File

MtsRdm

Description

Use a TMtsDataModule object as the central repository for all objects in a transactional application server that is written in Delphi. MTS data modules can contain any nonvisual components. . Typically these are data access components, such as datasets and the provider components that handle communication between these datasets and client applications. MTS data modules can also contain other nonvisual components, such as TTimer, or TImageList.

MTS data modules are dual-interface Automation servers that implement the IAppServer interface. Connection components on client applications are designed to look for this interface. The methods in this interface, however, do not call the SetComplete (see page 2410) method, so the MTS data module is not transactional. You may want to expand the MTS data module's interface to provide alternate versions of these methods that use the MTS data module's transaction.

The default interaction between a client dataset and an MTS data module does not rely on state information, so it is safe to use the MTS data module with just-in-time activation. However, you may need to provide a mechanism for communicating persistent state information to take advantage of some features such as incremental fetches.

To create an MTS data module at design time, choose File|New|Other, and select Transactional Data Module from the multitier page of the new items dialog. If you are writing the application server in Delphi, the Transactional Data Module Wizard then creates a new descendant of TMtsDataModule that supports an interface that descends from IAppServer.

See Also

IAppServer

Writing Applications Using COM

41.1.1.1 TMtsDataModule Members

The following tables list the members exposed by TMtsDataModule.

Protected Methods

	Name	Description
	Activate (see page 2400)	This is Activate, a member of class TMtsDataModule.
	AS_ApplyUpdates (see page 2401)	Applies updates received from a client dataset using a specified provider.
	AS_DataRequest (see page 2401)	Generates an OnAS_DataRequest event on the specified provider component.
	AS_Execute (see page 2402)	AS_Executes the query or stored procedure bound to a specified provider.
	AS_GetParams (see page 2403)	Fetches current parameter values from the dataset bound to a specified provider.
	AS_GetProviderNames (see page 2404)	Returns a list of all the available providers on the remote data module.
	AS_GetRecords (see page 2404)	Returns a data packet that contains the specified data.
	AS_RowRequest (see page 2405)	Returns information from a specified record of the provider's dataset.
V	CallComplete (see page 2406)	This is CallComplete, a member of class TMtsDataModule.
V	CanBePooled (see page 2406)	This is CanBePooled, a member of class TMtsDataModule.
	Deactivate (see page 2406)	This is Deactivate, a member of class TMtsDataModule.

Protected Properties

	Name	Description
R	ObjectContext (see page 2411)	This is ObjectContext, a member of class TMtsDataModule.

Public Methods

	Name	Description
V	Create (see page 2399)	Creates an MTS data module instance.
	DisableCommit (see page 2407)	Prevents clients from completing transactions.
	EnableCommit (see page 2407)	Allows clients to complete transactions without releasing state information.
	IsCallerInRole (see page 2408)	Indicates whether the current client fits into a particular security role.
	IsInTransaction (see page 2408)	Indicates whether the MTS data module is running in a transaction.
	IsSecurityEnabled (see page 2409)	Indicates whether security is enabled for the current MTS data module instance.
	SetAbort (see page 2409)	Allows the MTS data module to be deactivated and aborts any current transactions.
	SetComplete (see page 2410)	Indicates that the MTS data module has successfully completed its current task.

Published Events

	Name	Description
	OnActivate (see page 2412)	Occurs when an instance of TMtsDataModule (see page 2398) is activated.
	OnDeactivate (see page 2412)	Occurs when an instance of TMtsDataModule (see page 2398) is deactivated.

Published Properties

	Name	Description
	AutoComplete (see page 2411)	Specifies whether the IAppServer methods are transactional.
	Pooled (see page 2412)	Indicates whether the data module can be pooled when installed under COM+.

41.1.1.2 TMtsDataModule.Create Constructor

Creates an MTS data module instance.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TMtsDataModule(TComponent * AOwner);
```

Description

Most applications do not need to call the constructor for TMtsDataModule (see page 2398). Instead, add a remote data module to a server application using the transactional data module wizard on the multi-tier page of the new items dialog.

AOwner specifies the component that is responsible for freeing the remote data module (typically the application).

See Also

Create

41.1.1.3 TMtsDataModule Methods

The methods of the TMtsDataModule class are listed here.

Protected Methods

	Name	Description
	Activate (see page 2400)	This is Activate, a member of class TMtsDataModule.
	AS_ApplyUpdates (see page 2401)	Applies updates received from a client dataset using a specified provider.
	AS_DataRequest (see page 2401)	Generates an OnAS_DataRequest event on the specified provider component.
	AS_Execute (see page 2402)	AS_Executes the query or stored procedure bound to a specified provider.
	AS_GetParams (see page 2403)	Fetches current parameter values from the dataset bound to a specified provider.
	AS_GetProviderNames (see page 2404)	Returns a list of all the available providers on the remote data module.
	AS_GetRecords (see page 2404)	Returns a data packet that contains the specified data.
	AS_RowRequest (see page 2405)	Returns information from a specified record of the provider's dataset.
	CallComplete (see page 2406)	This is CallComplete, a member of class TMtsDataModule.
	CanBePooled (see page 2406)	This is CanBePooled, a member of class TMtsDataModule.
	Deactivate (see page 2406)	This is Deactivate, a member of class TMtsDataModule.

Public Methods

	Name	Description
	DisableCommit (see page 2407)	Prevents clients from completing transactions.
	EnableCommit (see page 2407)	Allows clients to complete transactions without releasing state information.
	IsCallerInRole (see page 2408)	Indicates whether the current client fits into a particular security role.
	IsInTransaction (see page 2408)	Indicates whether the MTS data module is running in a transaction.
	IsSecurityEnabled (see page 2409)	Indicates whether security is enabled for the current MTS data module instance.
	SetAbort (see page 2409)	Allows the MTS data module to be deactivated and aborts any current transactions.
	SetComplete (see page 2410)	Indicates that the MTS data module has successfully completed its current task.

41.1.1.3.1 TMtsDataModule.Activate Method

Pascal

```
procedure Activate; safecall;
```

C++

```
__fastcall Activate();
```

Description

This is Activate, a member of class TMtsDataModule.

41.1.1.3.2 TMtsDataModule.AS_ApplyUpdates Method

Applies updates received from a client dataset using a specified provider.

Pascal

```
function AS_ApplyUpdates(const ProviderName: WideString; Delta: OleVariant; MaxErrors:
Integer; out ErrorCount: Integer; var OwnerData: OleVariant): OleVariant; safecall;
```

C++

```
__fastcall OleVariant AS_ApplyUpdates(const BSTR ProviderName, OleVariant Delta, int
MaxErrors, int ErrorCount, OleVariant OwnerData);
```

Description

AS_AS_ApplyUpdates writes client updates to a dataset or database server. ProviderName specifies the name of the provider component that manages the update operation. Delta is a Variant containing the updated, inserted, and deleted records to write to the database. MaxErrors indicates the maximum number of errors to permit before rolling back the update operation; ErrorCount returns the actual number of errors encountered during the update operation. OwnerData is information supplied by the client application in a BeforeAS_ApplyUpdates event handler and returned to the client application in an AfterAS_ApplyUpdates event handler. It is passed to the provider's BeforeAS_ApplyUpdates event handler and returned from the provider's AfterAS_ApplyUpdates event handler.

AS_AS_ApplyUpdates The AS_ApplyUpdates_result parameter returns a Variant that is a data packet containing all records that could not be applied.

Note: Applications can only call the protected AS_AS_ApplyUpdates method using the TRemoteDataModule interface. To apply updates directly in code from the application server, use the provider component's AS_ApplyUpdates method.

See Also

ApplyUpdates

ApplyUpdates (see page 3100)

BeforeApplyUpdates (see page 3120)

AfterApplyUpdates (see page 3116)

BeforeApplyUpdates

AfterApplyUpdates

Providers

IAppServer

41.1.1.3.3 TMtsDataModule.AS_DataRequest Method

Generates an OnAS_DataRequest event on the specified provider component.

Pascal

```
function AS_DataRequest(const ProviderName: WideString; Data: OleVariant): OleVariant;
safecall;
```

C++

```
__fastcall OleVariant AS_DataRequest(const BSTR ProviderName, OleVariant Data);
```

Description

Use AS_AS_DataRequest to invoke an OnAS_DataRequest event handler of the provider named by ProviderName. This

method is provided to allow application developers to customize the communication between a client application and a provider component. There is no predefined meaning for the Data parameter or the return value AS_DataRequest_result parameter, which is returned to the caller.

Note: Applications can only call the protected AS_AS_DataRequest method using the TRemoteDataModule interface.

See Also

OnDataRequest (see page 3124)

Providers

DataRequest

IAppServer

41.1.1.3.4 TMtsDataModule.AS_Execute Method

AS_Executes the query or stored procedure bound to a specified provider.

Pascal

```
procedure AS_Execute(const ProviderName: WideString; const CommandText: WideString; var
Params: OleVariant; var OwnerData: OleVariant); safecall;
```

C++

```
__fastcall AS_Execute(const BSTR ProviderName, const BSTR CommandText, OleVariant Params,
OleVariant OwnerData);
```

Description

Use AS_Execute to pass an execute command to a remote provider, using a specified set of parameters. AS_Execute calls the AS_Execute method of the specified provider after assigning any parameters. Output parameters that result from executing a query or stored procedure are returned in the Params parameter.

Note: AS_AS_Execute should only be called for a provider whose dataset represents a query or stored procedure that does not return a cursor.

The ProviderName parameter specifies the name of the provider component associated with the query or stored procedure.

CommandText specifies an optional SQL statement that replaces the SQL of the provider's query, or the name of a table or stored procedure that replaces the provider's underlying database table or stored procedure. This parameter is ignored if the Options property does not include poAllowCommandText.

The Params parameter encodes any parameters expected by the query or stored procedure and returns any output parameters. To create a properly encoded value for Params from a TParams object, use the PackageParams method.

The OwnerData parameter contains custom information that appears as an argument to the provider's BeforeAS_Execute and AfterAS_Execute event handlers. This information originates in a client dataset's BeforeAS_Execute event handler. The value returned in OwnerData is passed to the client dataset's AfterAS_Execute event handler.

Note: Applications can only call the protected AS_AS_Execute method using the TRemoteDataModule interface. To execute a dataset in code from the application server, use the dataset component directly.

See Also

DataSet (see page 3160)

BeforeExecute (see page 3121)

AfterExecute (see page 3117)

Params

Execute

Execute (see page 3107)

Providers

IAppServer

BeforeExecute

AfterExecute

PackageParams

Options (see page 3087)

41.1.1.3.5 TMtsDataModule.AS_GetParams Method

Fetches current parameter values from the dataset bound to a specified provider.

Pascal

```
function AS_GetParams(const ProviderName: WideString; var OwnerData: OleVariant): OleVariant;
safecall;
```

C++

```
__fastcall OleVariant AS_GetParams(const BSTR ProviderName, OleVariant OwnerData);
```

Description

Use AS_GetParams to retrieve the current parameter values from the dataset associated with a provider. The ProviderName parameter specifies the name of the provider component whose dataset has the parameters. The OwnerData represents custom information that originates in a client dataset's BeforeGetParams event handler and returns information that is passed to the client dataset's AfterGetParams event handler.

GetParams_resultAS_GetParams returns the parameters, encoded as a Variant array, or, if the provider does not support parameter fetching, GetParams_resultAS_GetParams returns a Null Variant. Use the UnpackParams procedure to convert parameter values returned by AS_GetParams to a TParams object.

AS_GetParams should not be used to retrieve output parameters from stateless application servers, because parameter values may be changed by other applications. When writing a stateless application server, obtain output parameters using the AS_Execute method handler instead.

Note: Applications can only call the protected AS_GetParams method using the TRemoteDataModule interface.

See Also

DataSet (see page 3160)

Params

FetchParams

GetParams (see page 3108)

BeforeGetParams

AfterGetParams

Providers

IAppServer

AS_Execute

UnpackParams

41.1.1.3.6 TMtsDataModule.AS_GetProviderNames Method

Returns a list of all the available providers on the remote data module.

Pascal

```
function AS_GetProviderNames: OleVariant; safecall;
```

C++

```
__fastcall OleVariant AS_GetProviderNames();
```

Description

Call AS_AS_GetProviderNames for a list of all registered providers available from a remote data module. These values indicate the possible values for the ProviderName property of a client dataset that connects to the remote data module. When calling any of the remote data modules through the TRemoteDataModule interface, pass one of these names to indicate the provider that is the target of the method call.

AS_AS_GetProviderNames returns the names of the providers in a Variant array. Client datasets are associated with a provider by setting their ProviderName property to one of these names.

Note: Applications can only call the protected AS_AS_GetProviderNames method using the TRemoteDataModule interface.

See Also

ProviderName

Providers

IAppServer

Exported (see page 3115)

41.1.1.3.7 TMtsDataModule.AS_GetRecords Method

Returns a data packet that contains the specified data.

Pascal

```
function AS_GetRecords(const ProviderName: WideString; Count: Integer; out RecsOut: Integer;
Options: Integer; const CommandText: WideString; var Params: OleVariant; var OwnerData:
OleVariant): OleVariant; safecall;
```

C++

```
__fastcall OleVariant AS_GetRecords(const BSTR ProviderName, int Count, int RecsOut, int
Options, const BSTR CommandText, OleVariant Params, OleVariant OwnerData);
```

Description

Call AS_AS_GetRecords to retrieve a specified number of records. AS_AS_GetRecords returns the requested records, starting with the current record of the provider's dataset. When working with a stateless remote data module, you may need to reposition the cursor or re-execute a query or stored procedure in the provider's BeforeAS_GetRecords event handler.

ProviderName gives the name of the provider component that provides the records.

Count indicates the number or type of records to retrieve. If Count is -1, all records are retrieved. If Count is 0, only metadata is retrieved. If Count is greater than 0, only Count records are retrieved.

RecsOut returns the actual number of records retrieved.

Options indicates what information should be added to the data packet in addition to data. It is a combination of the

GetRecordOption constants. For example, the following value requests metadata along with a data packet encoded in XML:

```
MetaDataOption + XMLOption
MetaDataOption || XMLOption
```

CommandText specifies an optional SQL statement that replaces the SQL of the provider's query, or the name of a table or stored procedure that replaces the provider's underlying database table or stored procedure. This parameter is ignored if the provider's Options property does not include poAllowCommandText.

Params is a Variant containing any parameters that should be passed to the provider's dataset before it executes to generate the requested data. It returns any output parameters.

OwnerData contains custom information that is supplied by a client dataset's BeforeAS_GetRecords event handler. This information is passed to the provider's BeforeAS_GetRecords event handler. OwnerData returns information supplied by the provider's AfterAS_GetRecords event handler.

Records are returned as a data packet in a Variant parameter, AS_GetRecords_result.

Note: Applications can only call the protected AS_AS_GetRecords method using the TRemoteDataModule interface. To fetch records directly in code from the application server, use the provider component's AS_GetRecords method.

See Also

Data

Params

GetNextPacket

GetRecords (see page 3109)

BeforeGetRecords (see page 3122)

AfterGetRecords (see page 3118)

Providers

IAppServer

41.1.1.3.8 TMtsDataModule.AS_RowRequest Method

Returns information from a specified record of the provider's dataset.

Pascal

```
function AS_RowRequest(const ProviderName: WideString; Row: OleVariant; RequestType: Integer;
var OwnerData: OleVariant): OleVariant; safecall;
```

C++

```
__fastcall OleVariant AS_RowRequest(const BSTR ProviderName, OleVariant Row, int RequestType,
OleVariant OwnerData);
```

Description

Client datasets call AS_AS_RowRequest automatically to implement their FetchBlobs, FetchDetails , or RefreshRecord method.

The ProviderName parameter indicates the provider associated with the dataset from which information should be fetched.

The Row parameter is an OleVariant that describes the current record on the client dataset.

The RequestType parameter indicates the type of information required. It is an integer version of the TFetchOptions type. (An integer because the value may be marshaled to a remote application server). To create a value for RequestType, take the corresponding TFetchOptions value, cast it to a Byte, and cast the result to an Integer:

```
Integer(Byte([foBlobs, foDetails]));
```

```
(int) (Byte) (TFetchOptions() << foBlobs << foDetails);
```

OwnerData contains custom information that is supplied by a client dataset's BeforeAS_RowRequest event handler. This information is passed to the provider's BeforeAS_RowRequest event handler. OwnerData returns custom information supplied by the provider's AfterAS_RowRequest event handler.

The requested data is returned as a delta packet.

Note: Applications can only call the protected AS_AS_RowRequest method using the TRemoteDataModule interface.

See Also

RowRequest (see page 3113)

BeforeRowRequest (see page 3123)

AfterRowRequest (see page 3119)

FetchBlobs

FetchDetails

RefreshRecord

Providers

IAppServer

41.1.1.3.9 TMtsDataModule.CallComplete Method

Pascal

```
procedure CallComplete(Complete: Boolean); virtual;
```

C++

```
virtual __fastcall CallComplete(Boolean Complete);
```

Description

This is CallComplete, a member of class TMtsDataModule.

41.1.1.3.10 TMtsDataModule.CanBePooled Method

Pascal

```
function CanBePooled: Bool; virtual; stdcall;
```

C++

```
virtual __fastcall __stdcall Bool CanBePooled();
```

Description

This is CanBePooled, a member of class TMtsDataModule.

41.1.1.3.11 TMtsDataModule.Deactivate Method

Pascal

```
procedure Deactivate; stdcall;
```

C++

```
__fastcall __stdcall Deactivate();
```

Description

This is Deactivate, a member of class TMtsDataModule.

41.1.1.3.12 TMtsDataModule.DisableCommit Method

Prevents clients from completing transactions.

Pascal

```
procedure DisableCommit;
```

C++

```
__fastcall DisableCommit();
```

Description

Call DisableCommit to indicate that the MTS data module can't complete its current transaction or release state information (properties) until it receives further method invocations from the client. After calling DisableCommit, the current transaction (if any) can't be completed until the MTS data module calls EnableCommit (see page 2407) or SetComplete (see page 2410).

With transactions that are started automatically when the MTS data module's interface is called, DisableCommit is not necessary. These automatically generated transactions are not committed until the MTS data module calls SetComplete (see page 2410). However, when the MTS data module is enlisted in a transaction that was started previously, the client controls the transaction and can attempt to commit it before the MTS data module has completed its part. Calling DisableCommit prevents the client from prematurely committing a transaction. If a client tries to commit a transaction after the MTS data module has called DisableCommit, the transaction aborts.

Tip: Call DisableCommit at the beginning of a method to protect your MTS data module from committing prematurely in the event of an unexpected return. If the method call finishes normally, calling EnableCommit (see page 2407) or SetComplete (see page 2410) reverses this affect and allows the transaction to complete.

See Also

EnableCommit (see page 2407)

SetComplete (see page 2410)

SetAbort (see page 2409)

IsInTransaction (see page 2408)

41.1.1.3.13 TMtsDataModule.EnableCommit Method

Allows clients to complete transactions without releasing state information.

Pascal

```
procedure EnableCommit;
```

C++

```
__fastcall EnableCommit();
```

Description

Call EnableCommit to indicate that the MTS data module can complete its current transaction without releasing state information (properties). Unlike the SetComplete (see page 2410) method (which also signals that the current transaction can be committed), EnableCommit does not allow the data module to be deactivated. This lets the MTS data module preserve state information over multiple interface calls, while still allowing clients that have enlisted the data module in a transaction to commit.

By default, transactions can be completed when interface calls finish, unless the DisableCommit (see page 2407) method is

called. There is no need to call EnableCommit unless the MTS data module has previously called DisableCommit (see page 2407).

Note: Although EnableCommit permits the current transaction to be completed, it does not automatically commit the transaction. The client that enlisted the MTS data module in a transaction must still explicitly commit that transaction. If the transaction was started automatically (because the MTS data module requires a transaction for its interface calls), the transaction is not committed until the data module calls SetComplete (see page 2410).

See Also

DisableCommit (see page 2407)

SetComplete (see page 2410)

SetAbort (see page 2409)

IsInTransaction (see page 2408)

41.1.1.3.14 TMtsDataModule.IsCallerInRole Method

Indicates whether the current client fits into a particular security role.

Pascal

```
function IsCallerInRole(const Role: WideString): Bool;
```

C++

```
__fastcall Bool IsCallerInRole(const BSTR Role);
```

Description

MTS provides role-based security. Each level of access to the application server can be assigned a name, called a role. Using the MTS Explorer or COM+ administrator, administrators can prevent clients from using an interface based on their role. (This is called declarative security.) In addition, however, programs can provide more fine-grained security programmatically. To do this, use IsCallerInRole to provide access to blocks of code based on the caller's security role.

IsCallerInRole checks whether the current client is part of the role named by the Role parameter. It returns true if the client fits the named role, and false otherwise. Use IsCallerInRole to prevent clients that do not have the appropriate security roles from invoking certain blocks of code.

Note: MTS security is not available on servers running Windows95. If the application server is running on Windows95, IsCallerInRole always returns true. In addition, IsCallerInRole always returns true if the application server is part of an MTS Library package (which runs in the client's process space).

See Also

IsSecurityEnabled (see page 2409)

Using Transactional Data Modules

41.1.1.3.15 TMtsDataModule.IsInTransaction Method

Indicates whether the MTS data module is running in a transaction.

Pascal

```
function IsInTransaction: Bool;
```

C++

```
__fastcall Bool IsInTransaction();
```

Description

Call IsInTransaction to determine whether the MTS data module is currently running in a transaction. For example, if the MTS data module requires a transaction, use IsInTransaction to check that this requirement has not been turned off using the MTS Explorer or COM+ administrator, and to raise an exception if it has.

Note: If IsInTransaction returns false, the EnableCommit (see page 2407) and DisableCommit (see page 2407) methods have no effect.

See Also

EnableCommit (see page 2407)

DisableCommit (see page 2407)

SetComplete (see page 2410)

SetAbort (see page 2409)

41.1.1.3.16 TMtsDataModule.IsSecurityEnabled Method

Indicates whether security is enabled for the current MTS data module instance.

Pascal

```
function IsSecurityEnabled: Bool;
```

C++

```
__fastcall Bool IsSecurityEnabled();
```

Description

Call IsSecurityEnabled to determine whether the MTS data module is currently protected by MTS or COM+ security services. MTS or COM+ security is enabled unless the object is running in the client's process. This can happen if an administrator configures the MTS data module to run in the client's process space (for example, an MTS Library package), or if the client application is in the same MTS package as the MTS data module.

Note: MTS security is not available for servers running on Windows95.

See Also

IsCallerInRole (see page 2408)

41.1.1.3.17 TMtsDataModule.SetAbort Method

Allows the MTS data module to be deactivated and aborts any current transactions.

Pascal

```
procedure SetAbort;
```

C++

```
__fastcall SetAbort();
```

Description

Call SetAbort to indicate that any current transaction that includes the MTS data module should abort. If MTS or COM+ started a transaction automatically (because the data module requires one), that transaction aborts when the current method call exits. If the MTS data module was enlisted in a pre-existing transaction, the client transaction will eventually fail. Usually, SetAbort is called in the except block of a try..exceptstatement.

In addition to aborting any current transaction, SetAbort indicates that the MTS data module can be deactivated when the current

interface call exits. This means that state information is not preserved for the next time the client calls the data module's interface. To prevent transactions from committing successfully without losing state information, use the DisableCommit (see page 2407) method instead.

Tip: Call SetAbort at the beginning of a method to protect your MTS data module from committing prematurely in the event of an unexpected return. If the method call finishes normally, calling SetComplete (see page 2410) reverses this affect and allows the transaction to complete.

See Also

SetComplete (see page 2410)

DisableCommit (see page 2407)

EnableCommit (see page 2407)

IsInTransaction (see page 2408)

41.1.1.3.18 TMtsDataModule.SetComplete Method

Indicates that the MTS data module has successfully completed its current task.

Pascal

```
procedure SetComplete;
```

C++

```
__fastcall SetComplete();
```

Description

Call SetComplete to indicate that the MTS data module no longer needs to maintain state information. After calling SetComplete, the MTS data module is deactivated when the current method call exits.

If the MTS data module is part of a transaction, SetComplete indicates that this part of the transaction can commit. If MTS or COM+ started the transaction automatically when the client called the MTS data module's method (because the MTS data module requires a transaction), it tries to commit the transaction when the current method exits.

Note: This transaction may still fail if the current transaction included another MTS or COM+ object that called SetAbort (see page 2409) or DisableCommit (see page 2407).

See Also

SetAbort (see page 2409)

DisableCommit (see page 2407)

EnableCommit (see page 2407)

IsInTransaction (see page 2408)

Extending the Interface of the Application Server

41.1.1.4 TMtsDataModule Properties

The properties of the TMtsDataModule class are listed here.

Protected Properties

	Name	Description
	ObjectContext (see page 2411)	This is ObjectContext, a member of class TMtsDataModule.

Published Properties

	Name	Description
	AutoComplete (see page 2411)	Specifies whether the IAppServer methods are transactional.
	Pooled (see page 2412)	Indicates whether the data module can be pooled when installed under COM+.

41.1.1.4.1 TMtsDataModule.AutoComplete Property

Specifies whether the IAppServer methods are transactional.

Pascal

```
property AutoComplete: Boolean;
```

C++

```
__property Boolean AutoComplete;
```

Description

AutoComplete controls whether calls made using the IAppServer interface end in a call to SetComplete (see page 2410) or SetAbort (see page 2409). By calling SetComplete (see page 2410) or SetAbort (see page 2409), the MTS data module indicates that no state information needs to be preserved when the call exits. This allows any MTS transactions to complete (or roll back) and allows COM+ or the MTS proxy to deactivate the MTS data module.

When AutoComplete is true, all IAppServer methods call SetComplete (see page 2410) when they are done with two exceptions:

AS_ApplyUpdates (see page 2401) calls SetAbort (see page 2409) if an exception is raised or if the error count exceeds the specified maximum.

AS_DataRequest (see page 2401) doesn't call either SetComplete (see page 2410) or SetAbort (see page 2409). You must add the appropriate call to the OnDataRequest event handler.

Set AutoComplete to false if the MTS data module needs to preserve state information (such as the current record in a dataset) after an IAppServer call exits. When AutoComplete is false, you must add your own interface methods that call SetComplete (see page 2410) or SetAbort (see page 2409) when the MTS data module can be safely deactivated.

See Also

IAppServer

SetComplete (see page 2410)

SetAbort (see page 2409)

41.1.1.4.2 TMtsDataModule.ObjectContext Property

Pascal

```
property ObjectContext: IObjectContext;
```

C++

```
__property IObjectContext ObjectContext;
```

Description

This is ObjectContext, a member of class TMtsDataModule.

41.1.1.4.3 TMtsDataModule.Pooled Property

Indicates whether the data module can be pooled when installed under COM+.

Pascal

```
property Pooled: Boolean;
```

C++

```
__property Boolean Pooled;
```

Description

Set Pooled to indicate whether instances of the MTS data module can be pooled when installed under COM+. The MTS remote data module uses the Pooled property when implementing the IObjectControl (see page 2424) method CanBePooled (see page 2406). This instructs COM+ that the object can be moved to an object pool when deactivated, so that the single instance is shared by all clients.

Warning: Do not set Pooled to True unless the MTS data module is stateless.

41.1.1.5 TMtsDataModule Events

The events of the TMtsDataModule class are listed here.

Published Events

	Name	Description
	OnActivate (see page 2412)	Occurs when an instance of TMtsDataModule (see page 2398) is activated.
	OnDeactivate (see page 2412)	Occurs when an instance of TMtsDataModule (see page 2398) is deactivated.

41.1.1.5.1 TMtsDataModule.OnActivate Event

Occurs when an instance of TMtsDataModule (see page 2398) is activated.

Pascal

```
property OnActivate: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnActivate;
```

Description

Write an OnActivate event handler to take specific action when COMN+ or the MTS proxy activates the MTS data module. This is the point when any database connections should be established. Database connections must not opened before the MTS data module is activated.

Note: In traditional MTS or COM+ programming, objects acquire their context object when they are activated. However, TMtsDataModule (see page 2398) handles this task automatically.

See Also

OnDeactivate (see page 2412)

41.1.1.5.2 TMtsDataModule.OnDeactivate Event

Occurs when an instance of TMtsDataModule (see page 2398) is deactivated.

Pascal

```
property OnDeactivate: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnDeactivate;
```

Description

Write an OnDeactivate event handler to take specific action when COM+ or the MTS proxy releases an instance of the MTS data module. Use this method to close any database connections that were opened when the MTS data module was activated. Database connections must not be left open after the MTS data module is deactivated.

Note: In traditional COM+ and MTS programming, objects release their context object when they are deactivated. However, TMtsDataModule (see page 2398) handles this task automatically.

See Also

OnActivate (see page 2412)

42 Mtx Namespace

This is namespace Mtx.

Topics

Name	Description
Interfaces (see page 2417)	The following table lists interfaces in this documentation.
Functions (see page 2437)	The following table lists functions in this documentation.
Constants (see page 2442)	The following table lists constants in this documentation.

Constants

Name	Description
CLASS_SharedPropertyGroupManager (see page 2442)	This is constant Mtx.CLASS_SharedPropertyGroupManager.
CLASS_TransactionContext (see page 2442)	This is constant Mtx.CLASS_TransactionContext.
CLASS_TransactionContextEx (see page 2443)	This is constant Mtx.CLASS_TransactionContextEx.
LockMethod (see page 2443)	This is constant Mtx.LockMethod.
LockSetGet (see page 2443)	This is constant Mtx.LockSetGet.
mtsErrCtxAborted (see page 2444)	This is constant Mtx.mtsErrCtxAborted.
mtsErrCtxAborting (see page 2444)	This is constant Mtx.mtsErrCtxAborting.
mtsErrCtxActivityTimeout (see page 2444)	This is constant Mtx.mtsErrCtxActivityTimeout.
mtsErrCtxNoContext (see page 2445)	This is constant Mtx.mtsErrCtxNoContext.
mtsErrCtxNoSecurity (see page 2445)	This is constant Mtx.mtsErrCtxNoSecurity.
mtsErrCtxNotRegistered (see page 2445)	This is constant Mtx.mtsErrCtxNotRegistered.
mtsErrCtxOldReference (see page 2446)	This is constant Mtx.mtsErrCtxOldReference.
mtsErrCtxRoleNotFound (see page 2446)	This is constant Mtx.mtsErrCtxRoleNotFound.
mtsErrCtxTMNotAvailable (see page 2446)	This is constant Mtx.mtsErrCtxTMNotAvailable.
mtsErrCtxWrongThread (see page 2447)	This is constant Mtx.mtsErrCtxWrongThread.
Process (see page 2447)	This is constant Mtx.Process.
Standard (see page 2447)	This is constant Mtx.Standard.

Functions

	Name	Description
	CreateSharedPropertyGroup (see page 2437)	Returns the interface for a SharedPropertyGroup that allows MTS objects within a server process to share state.
	CreateSharedPropertyGroupManager (see page 2438)	Returns an interface for a new shared property group manager that creates and coordinates shared properties of MTS objects.
	CreateTransactionContext (see page 2438)	Starts an MTS transaction.

	CreateTransactionContextEx (see page 2439)	Instantiates a TransactionContextEx object, which marks the beginning of a transaction.
	GetObjectContext (see page 2440)	Fetches the IObjectContext (see page 2420) interface for the current thread.
	SafeRef (see page 2440)	Returns a reference to a transactional object that is safe to pass outside its context.

Interfaces

	Name	Description
	IEnumNames (see page 2417)	This is class Mtx.IEnumNames.
	IGetContextProperties (see page 2419)	This is class Mtx.IGetContextProperties.
	IObjectContext (see page 2420)	This is class Mtx.IObjectContext.
	IObjectContextActivity (see page 2423)	This is class Mtx.IObjectContextActivity.
	IObjectControl (see page 2424)	This is class Mtx.IObjectControl.
	ISecurityProperty (see page 2425)	This is class Mtx.ISecurityProperty.
	ISharedProperty (see page 2427)	This is class Mtx.ISharedProperty.
	ISharedPropertyGroup (see page 2429)	This is class Mtx.ISharedPropertyGroup.
	ISharedPropertyGroupManager (see page 2431)	This is class Mtx.ISharedPropertyGroupManager.
	ITransactionContext (see page 2433)	This is class Mtx.ITransactionContext.
	ITransactionContextEx (see page 2435)	This is class Mtx.ITransactionContextEx.

42.1 Interfaces

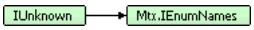
The following table lists interfaces in this documentation.

Interfaces

	Name	Description
	IEnumNames (see page 2417)	This is class Mtx.IEnumNames.
	IGetContextProperties (see page 2419)	This is class Mtx.IGetContextProperties.
	IObjectContext (see page 2420)	This is class Mtx.IObjectContext.
	IObjectContextActivity (see page 2423)	This is class Mtx.IObjectContextActivity.
	IObjectControl (see page 2424)	This is class Mtx.IObjectControl.
	ISecurityProperty (see page 2425)	This is class Mtx.ISecurityProperty.
	ISharedProperty (see page 2427)	This is class Mtx.ISharedProperty.
	ISharedPropertyGroup (see page 2429)	This is class Mtx.ISharedPropertyGroup.
	ISharedPropertyGroupManager (see page 2431)	This is class Mtx.ISharedPropertyGroupManager.
	ITransactionContext (see page 2433)	This is class Mtx.ITransactionContext.
	ITransactionContextEx (see page 2435)	This is class Mtx.ITransactionContextEx.

42.1.1 IEnumNames Interface

Class Hierarchy



Pascal

```
IEnumNames = interface(IUnknown);
```

C++

```
__interface IEnumNames : public IUnknown;
```

File

Mtx

Description

This is class Mtx.IEnumNames.

42.1.1.1 IEnumNames Members

The following tables list the members exposed by IEnumNames.

Public Methods

	Name	Description
	Clone (see page 2418)	This is Clone, a member of class IEnumNames.
	Next (see page 2418)	This is Next, a member of class IEnumNames.
	Reset (see page 2418)	This is Reset, a member of class IEnumNames.
	Skip (see page 2418)	This is Skip, a member of class IEnumNames.

42.1.1.2 IEnumNames Methods

The methods of the IEnumNames class are listed here.

Public Methods

	Name	Description
	Clone (see page 2418)	This is Clone, a member of class IEnumNames.
	Next (see page 2418)	This is Next, a member of class IEnumNames.
	Reset (see page 2418)	This is Reset, a member of class IEnumNames.
	Skip (see page 2418)	This is Skip, a member of class IEnumNames.

42.1.1.2.1 IEnumNames.Clone Method

Pascal

```
function Clone(out enm: IEnumNames): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT Clone(IEnumNames enm);
```

Description

This is Clone, a member of class IEnumNames.

42.1.1.2.2 IEnumNames.Next Method

Pascal

```
function Next(celt: UINT; out rgname: WideString; out pceltFetched: UINT): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT Next(UINT celt, BSTR rgname, UINT pceltFetched);
```

Description

This is Next, a member of class IEnumNames.

42.1.1.2.3 IEnumNames.Reset Method

Pascal

```
function Reset: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT Reset();
```

Description

This is Reset, a member of class IEnumNames.

42.1.1.2.4 IEnumNames.Skip Method

Pascal

```
function Skip(celt: UINT): HResult; stdcall;
```

C++

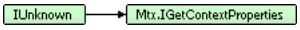
```
__fastcall __stdcall HRESULT Skip(UINT celt);
```

Description

This is Skip, a member of class IEnumNames.

42.1.2 IGetContextProperties Interface

Class Hierarchy



Pascal

```
IGetContextProperties = interface(IUnknown);
```

C++

```
__interface IGetContextProperties : public IUnknown;
```

File

Mtx

Description

This is class Mtx.IGetContextProperties.

42.1.2.1 IGetContextProperties Members

The following tables list the members exposed by IGetContextProperties.

Public Methods

	Name	Description
	Count (see page 2419)	This is Count, a member of class IGetContextProperties.
	EnumNames (see page 2420)	This is EnumNames, a member of class IGetContextProperties.
	GetProperty (see page 2420)	This is GetProperty, a member of class IGetContextProperties.

42.1.2.2 IGetContextProperties Methods

The methods of the IGetContextProperties class are listed here.

Public Methods

	Name	Description
	Count (see page 2419)	This is Count, a member of class IGetContextProperties.
	EnumNames (see page 2420)	This is EnumNames, a member of class IGetContextProperties.
	GetProperty (see page 2420)	This is GetProperty, a member of class IGetContextProperties.

42.1.2.2.1 IGetContextProperties.Count Method

Pascal

```
function Count: Integer; safecall;
```

C++

```
__fastcall int Count();
```

Description

This is Count, a member of class IGetContextProperties.

42.1.2.2.2 IGetContextProperties.EnumNames Method

Pascal

```
function EnumNames: IEnumNames; safecall;
```

C++

```
__fastcall IEnumNames EnumNames();
```

Description

This is EnumNames, a member of class IGetContextProperties.

42.1.2.2.3 IGetContextProperties.GetProperty Method

Pascal

```
function GetProperty(const name: WideString): OleVariant; safecall;
```

C++

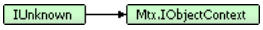
```
__fastcall OleVariant GetProperty(const BSTR name);
```

Description

This is GetProperty, a member of class IGetContextProperties.

42.1.3 IObjectContext Interface

Class Hierarchy



Pascal

```
IObjectContext = interface(IUnknown);
```

C++

```
__interface IObjectContext : public IUnknown;
```

File

Mtx

Description

This is class Mtx.IObjectContext.

42.1.3.1 IObjectContext Members

The following tables list the members exposed by IObjectContext.

Public Methods

	Name	Description
	CreateInstance (see page 2421)	This is CreateInstance, a member of class IObjectContext.
	DisableCommit (see page 2421)	This is DisableCommit, a member of class IObjectContext.
	EnableCommit (see page 2422)	This is EnableCommit, a member of class IObjectContext.
	IsCallerInRole (see page 2422)	This is IsCallerInRole, a member of class IObjectContext.
	IsInTransaction (see page 2422)	This is IsInTransaction, a member of class IObjectContext.
	IsSecurityEnabled (see page 2422)	This is IsSecurityEnabled, a member of class IObjectContext.
	SetAbort (see page 2422)	This is SetAbort, a member of class IObjectContext.
	SetComplete (see page 2423)	This is SetComplete, a member of class IObjectContext.

42.1.3.2 IObjectContext Methods

The methods of the IObjectContext class are listed here.

Public Methods

	Name	Description
	CreateInstance (see page 2421)	This is CreateInstance, a member of class IObjectContext.
	DisableCommit (see page 2421)	This is DisableCommit, a member of class IObjectContext.
	EnableCommit (see page 2422)	This is EnableCommit, a member of class IObjectContext.
	IsCallerInRole (see page 2422)	This is IsCallerInRole, a member of class IObjectContext.
	IsInTransaction (see page 2422)	This is IsInTransaction, a member of class IObjectContext.
	IsSecurityEnabled (see page 2422)	This is IsSecurityEnabled, a member of class IObjectContext.
	SetAbort (see page 2422)	This is SetAbort, a member of class IObjectContext.
	SetComplete (see page 2423)	This is SetComplete, a member of class IObjectContext.

42.1.3.2.1 IObjectContext.CreateInstance Method

Pascal

```
function CreateInstance(const cid: TGUID; const rid: TGUID; out pv): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT CreateInstance(const TGUID cid, const TGUID rid,  pv);
```

Description

This is CreateInstance, a member of class IObjectContext.

42.1.3.2.2 IObjectContext.DisableCommit Method

Pascal

```
procedure DisableCommit; safecall;
```

C++

```
__fastcall DisableCommit();
```

Description

This is DisableCommit, a member of class IObjectContext.

42.1.3.2.3 IObjectContext.EnableCommit Method

Pascal

```
procedure EnableCommit; safecall;
```

C++

```
__fastcall EnableCommit();
```

Description

This is EnableCommit, a member of class IObjectContext.

42.1.3.2.4 IObjectContext.IsCallerInRole Method

Pascal

```
function IsCallerInRole(const bstrRole: WideString): Bool; safecall;
```

C++

```
__fastcall Bool IsCallerInRole(const BSTR bstrRole);
```

Description

This is IsCallerInRole, a member of class IObjectContext.

42.1.3.2.5 IObjectContext.IsInTransaction Method

Pascal

```
function IsInTransaction: Bool; stdcall;
```

C++

```
__fastcall __stdcall Bool IsInTransaction();
```

Description

This is IsInTransaction, a member of class IObjectContext.

42.1.3.2.6 IObjectContext.IsSecurityEnabled Method

Pascal

```
function IsSecurityEnabled: Bool; stdcall;
```

C++

```
__fastcall __stdcall Bool IsSecurityEnabled();
```

Description

This is IsSecurityEnabled, a member of class IObjectContext.

42.1.3.2.7 IObjectContext.SetAbort Method

Pascal

```
procedure SetAbort; safecall;
```

C++

```
__fastcall SetAbort();
```

Description

This is SetAbort, a member of class IObjectContext.

42.1.3.2.8 IObjectContext.SetComplete Method

Pascal

```
procedure SetComplete; safecall;
```

C++

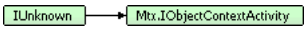
```
__fastcall SetComplete();
```

Description

This is SetComplete, a member of class IObjectContext.

42.1.4 IObjectContextActivity Interface

Class Hierarchy



Pascal

```
IObjectContextActivity = interface(IUnknown);
```

C++

```
__interface IObjectContextActivity : public IUnknown;
```

File

Mtx

Description

This is class Mtx.IObjectContextActivity.

42.1.4.1 IObjectContextActivity Members

The following tables list the members exposed by IObjectContextActivity.

Public Methods

	Name	Description
	GetActivityId (see page 2424)	This is GetActivityId, a member of class IObjectContextActivity.

42.1.4.2 IObjectContextActivity Methods

The methods of the IObjectContextActivity class are listed here.

Public Methods

	Name	Description
	GetActivityId (see page 2424)	This is GetActivityId, a member of class IObjectContextActivity.

42.1.4.2.1 IObjectContextActivity.GetActivityId Method

Pascal

```
function GetActivityId: TGuid; safecall;
```

C++

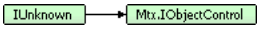
```
__fastcall TGuid GetActivityId();
```

Description

This is GetActivityId, a member of class IObjectContextActivity.

42.1.5 IObjectControl Interface

Class Hierarchy



Pascal

```
IObjectControl = interface(IUnknown);
```

C++

```
__interface IObjectControl : public IUnknown;
```

File

Mtx

Description

This is class Mtx.IObjectControl.

42.1.5.1 IObjectControl Members

The following tables list the members exposed by IObjectControl.

Public Methods

	Name	Description
	Activate (see page 2425)	This is Activate, a member of class IObjectControl.
	CanBePooled (see page 2425)	This is CanBePooled, a member of class IObjectControl.
	Deactivate (see page 2425)	This is Deactivate, a member of class IObjectControl.

42.1.5.2 IObjectControl Methods

The methods of the IObjectControl class are listed here.

Public Methods

	Name	Description
	Activate (see page 2425)	This is Activate, a member of class IObjectControl.
	CanBePooled (see page 2425)	This is CanBePooled, a member of class IObjectControl.
	Deactivate (see page 2425)	This is Deactivate, a member of class IObjectControl.

42.1.5.2.1 IObjectControl.Activate Method

Pascal

```
procedure Activate; safecall;
```

C++

```
__fastcall Activate();
```

Description

This is Activate, a member of class IObjectControl.

42.1.5.2.2 IObjectControl.CanBePooled Method

Pascal

```
function CanBePooled: Bool; stdcall;
```

C++

```
__fastcall __stdcall Bool CanBePooled();
```

Description

This is CanBePooled, a member of class IObjectControl.

42.1.5.2.3 IObjectControl.Deactivate Method

Pascal

```
procedure Deactivate; stdcall;
```

C++

```
__fastcall __stdcall Deactivate();
```

Description

This is Deactivate, a member of class IObjectControl.

42.1.6 ISecurityProperty Interface

Class Hierarchy

Pascal

```
ISecurityProperty = interface(IUnknown);
```

C++

```
__interface ISecurityProperty : public IUnknown;
```

File

Mtx

Description

This is class Mtx.ISecurityProperty.

42.1.6.1 ISecurityProperty Members

The following tables list the members exposed by ISecurityProperty.

Public Methods

	Name	Description
	GetDirectCallerSID (see page 2426)	This is GetDirectCallerSID, a member of class ISecurityProperty.
	GetDirectCreatorSID (see page 2426)	This is GetDirectCreatorSID, a member of class ISecurityProperty.
	GetOriginalCallerSID (see page 2426)	This is GetOriginalCallerSID, a member of class ISecurityProperty.
	GetOriginalCreatorSID (see page 2427)	This is GetOriginalCreatorSID, a member of class ISecurityProperty.
	ReleaseSID (see page 2427)	This is ReleaseSID, a member of class ISecurityProperty.

42.1.6.2 ISecurityProperty Methods

The methods of the ISecurityProperty class are listed here.

Public Methods

	Name	Description
	GetDirectCallerSID (see page 2426)	This is GetDirectCallerSID, a member of class ISecurityProperty.
	GetDirectCreatorSID (see page 2426)	This is GetDirectCreatorSID, a member of class ISecurityProperty.
	GetOriginalCallerSID (see page 2426)	This is GetOriginalCallerSID, a member of class ISecurityProperty.
	GetOriginalCreatorSID (see page 2427)	This is GetOriginalCreatorSID, a member of class ISecurityProperty.
	ReleaseSID (see page 2427)	This is ReleaseSID, a member of class ISecurityProperty.

42.1.6.2.1 ISecurityProperty.GetDirectCallerSID Method

Pascal

```
function GetDirectCallerSID(out sid: PSID): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetDirectCallerSID(PSID sid);
```

Description

This is GetDirectCallerSID, a member of class ISecurityProperty.

42.1.6.2.2 ISecurityProperty.GetDirectCreatorSID Method

Pascal

```
function GetDirectCreatorSID(out sid: PSID): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetDirectCreatorSID(PSID sid);
```

Description

This is GetDirectCreatorSID, a member of class ISecurityProperty.

42.1.6.2.3 ISecurityProperty.GetOriginalCallerSID Method

Pascal

```
function GetOriginalCallerSID(out sid: PSID): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetOriginalCallerSID(PSID sid);
```

Description

This is GetOriginalCallerSID, a member of class ISecurityProperty.

42.1.6.2.4 ISecurityProperty.GetOriginalCreatorSID Method

Pascal

```
function GetOriginalCreatorSID(out sid: PSID): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetOriginalCreatorSID(PSID sid);
```

Description

This is GetOriginalCreatorSID, a member of class ISecurityProperty.

42.1.6.2.5 ISecurityProperty.ReleaseSID Method

Pascal

```
function ReleaseSID(sid: PSID): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT ReleaseSID(PSID sid);
```

Description

This is ReleaseSID, a member of class ISecurityProperty.

42.1.7 ISharedProperty Interface

Class Hierarchy

Pascal

```
ISharedProperty = interface(IDispatch);
```

C++

```
__interface ISharedProperty : public IDispatch;
```

File

Mtx

Description

This is class Mtx.ISharedProperty.

42.1.7.1 ISharedProperty Members

The following tables list the members exposed by ISharedProperty.

Public Methods

	Name	Description
	Get_Value (see page 2428)	This is Get_Value, a member of class ISharedProperty.
	Set_Value (see page 2428)	This is Set_Value, a member of class ISharedProperty.

Public Properties

	Name	Description
	Value (see page 2429)	This is Value, a member of class ISharedProperty.

42.1.7.2 ISharedProperty Methods

The methods of the ISharedProperty class are listed here.

Public Methods

	Name	Description
	Get_Value (see page 2428)	This is Get_Value, a member of class ISharedProperty.
	Set_Value (see page 2428)	This is Set_Value, a member of class ISharedProperty.

42.1.7.2.1 ISharedProperty.Get_Value Method

Pascal

```
function Get_Value: OleVariant; safecall;
```

C++

```
__fastcall OleVariant Get_Value();
```

Description

This is Get_Value, a member of class ISharedProperty.

42.1.7.2.2 ISharedProperty.Set_Value Method

Pascal

```
procedure Set_Value(pVal: OleVariant); safecall;
```

C++

```
__fastcall Set_Value(OleVariant pVal);
```

Description

This is Set_Value, a member of class ISharedProperty.

42.1.7.3 ISharedProperty Properties

The properties of the ISharedProperty class are listed here.

Public Properties

	Name	Description
	Value (see page 2429)	This is Value, a member of class ISharedProperty.

42.1.7.3.1 ISharedProperty.Value Property

Pascal

```
property Value: OleVariant;
```

C++

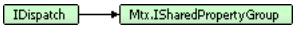
```
__property OleVariant Value;
```

Description

This is Value, a member of class ISharedProperty.

42.1.8 ISharedPropertyGroup Interface

Class Hierarchy



Pascal

```
ISharedPropertyGroup = interface(IDispatch);
```

C++

```
__interface ISharedPropertyGroup : public IDispatch;
```

File

Mtx

Description

This is class Mtx.ISharedPropertyGroup.

42.1.8.1 ISharedPropertyGroup Members

The following tables list the members exposed by ISharedPropertyGroup.

Public Methods

	Name	Description
	CreateProperty (see page 2430)	This is CreateProperty, a member of class ISharedPropertyGroup.
	CreatePropertyByPosition (see page 2430)	This is CreatePropertyByPosition, a member of class ISharedPropertyGroup.
	Get_PropertyByName (see page 2430)	This is Get_PropertyByName, a member of class ISharedPropertyGroup.
	Get_PropertyByPosition (see page 2430)	This is Get_PropertyByPosition, a member of class ISharedPropertyGroup.

Public Properties

	Name	Description
R	PropertyByName (see page 2431)	This is PropertyByName, a member of class ISharedPropertyGroup.
R	PropertyByPosition (see page 2431)	This is PropertyByPosition, a member of class ISharedPropertyGroup.

42.1.8.2 ISharedPropertyGroup Methods

The methods of the ISharedPropertyGroup class are listed here.

Public Methods

	Name	Description
	CreateProperty (see page 2430)	This is CreateProperty, a member of class ISharedPropertyGroup.
	CreatePropertyByPosition (see page 2430)	This is CreatePropertyByPosition, a member of class ISharedPropertyGroup.
	Get_PropertyByName (see page 2430)	This is Get_PropertyByName, a member of class ISharedPropertyGroup.
	Get_PropertyByPosition (see page 2430)	This is Get_PropertyByPosition, a member of class ISharedPropertyGroup.

42.1.8.2.1 ISharedPropertyGroup.CreateProperty Method

Pascal

```
function CreateProperty(const Name: WideString; out fExists: WordBool): ISharedProperty;
safecall;
```

C++

```
__fastcall ISharedProperty CreateProperty(const BSTR Name, WordBool fExists);
```

Description

This is CreateProperty, a member of class ISharedPropertyGroup.

42.1.8.2.2 ISharedPropertyGroup.CreatePropertyByPosition Method

Pascal

```
function CreatePropertyByPosition(Index: Integer; out fExists: WordBool): ISharedProperty;
safecall;
```

C++

```
__fastcall ISharedProperty CreatePropertyByPosition(int Index, WordBool fExists);
```

Description

This is CreatePropertyByPosition, a member of class ISharedPropertyGroup.

42.1.8.2.3 ISharedPropertyGroup.Get_PropertyByName Method

Pascal

```
function Get_PropertyByName(const Name: WideString): ISharedProperty; safecall;
```

C++

```
__fastcall ISharedProperty Get_PropertyByName(const BSTR Name);
```

Description

This is Get_PropertyByName, a member of class ISharedPropertyGroup.

42.1.8.2.4 ISharedPropertyGroup.Get_PropertyByPosition Method

Pascal

```
function Get_PropertyByPosition(Index: Integer): ISharedProperty; safecall;
```

C++

```
__fastcall ISharedProperty Get_PropertyByPosition(int Index);
```

Description

This is Get_PropertyByPosition, a member of class ISharedPropertyGroup.

42.1.8.3 ISharedPropertyGroup Properties

The properties of the ISharedPropertyGroup class are listed here.

Public Properties

	Name	Description
	PropertyByName (see page 2431)	This is PropertyByName, a member of class ISharedPropertyGroup.
	PropertyByPosition (see page 2431)	This is PropertyByPosition, a member of class ISharedPropertyGroup.

42.1.8.3.1 ISharedPropertyGroup.PropertyByName Property

Pascal

```
property PropertyByName [const Name: WideString]: ISharedProperty;
```

C++

```
__property ISharedProperty PropertyByName[BSTR const Name];
```

Description

This is PropertyByName, a member of class ISharedPropertyGroup.

42.1.8.3.2 ISharedPropertyGroup.PropertyByPosition Property

Pascal

```
property PropertyByPosition [Index: Integer]: ISharedProperty;
```

C++

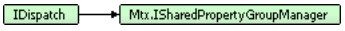
```
__property ISharedProperty PropertyByPosition[int Index];
```

Description

This is PropertyByPosition, a member of class ISharedPropertyGroup.

42.1.9 ISharedPropertyGroupManager Interface

Class Hierarchy



Pascal

```
ISharedPropertyGroupManager = interface(IDispatch);
```

C++

```
__interface ISharedPropertyGroupManager : public IDispatch;
```

File

Mtx

Description

This is class Mtx.ISharedPropertyGroupManager.

42.1.9.1 ISharedPropertyGroupManager Members

The following tables list the members exposed by ISharedPropertyGroupManager.

Public Methods

	Name	Description
	CreatePropertyGroup (see page 2432)	This is CreatePropertyGroup, a member of class ISharedPropertyGroupManager.
	Get__NewEnum (see page 2432)	This is Get__NewEnum, a member of class ISharedPropertyGroupManager.
	Get_Group (see page 2433)	This is Get_Group, a member of class ISharedPropertyGroupManager.

Public Properties

	Name	Description
R	_NewEnum (see page 2433)	This is _NewEnum, a member of class ISharedPropertyGroupManager.
R	Group (see page 2433)	This is Group, a member of class ISharedPropertyGroupManager.

42.1.9.2 ISharedPropertyGroupManager Methods

The methods of the ISharedPropertyGroupManager class are listed here.

Public Methods

	Name	Description
	CreatePropertyGroup (see page 2432)	This is CreatePropertyGroup, a member of class ISharedPropertyGroupManager.
	Get__NewEnum (see page 2432)	This is Get__NewEnum, a member of class ISharedPropertyGroupManager.
	Get_Group (see page 2433)	This is Get_Group, a member of class ISharedPropertyGroupManager.

42.1.9.2.1 ISharedPropertyGroupManager.CreatePropertyGroup Method

Pascal

```
function CreatePropertyGroup(const Name: WideString; var LockMode: Integer; var RelMode:
Integer; out fExists: WordBool): ISharedPropertyGroup; safecall;
```

C++

```
__fastcall ISharedPropertyGroup CreatePropertyGroup(const BSTR Name, int LockMode, int
RelMode, WordBool fExists);
```

Description

This is CreatePropertyGroup, a member of class ISharedPropertyGroupManager.

42.1.9.2.2 ISharedPropertyGroupManager.Get__NewEnum Method

Pascal

```
function Get__NewEnum: IUnknown; safecall;
```

C++

```
__fastcall IUnknown Get__NewEnum();
```

Description

This is Get__NewEnum, a member of class ISharedPropertyGroupManager.

42.1.9.2.3 ISharedPropertyGroupManager.Get_Group Method

Pascal

```
function Get_Group(const Name: WideString): ISharedPropertyGroup; safecall;
```

C++

```
__fastcall ISharedPropertyGroup Get_Group(const BSTR Name);
```

Description

This is Get_Group, a member of class ISharedPropertyGroupManager.

42.1.9.3 ISharedPropertyGroupManager Properties

The properties of the ISharedPropertyGroupManager class are listed here.

Public Properties

	Name	Description
	_NewEnum (see page 2433)	This is _NewEnum, a member of class ISharedPropertyGroupManager.
	Group (see page 2433)	This is Group, a member of class ISharedPropertyGroupManager.

42.1.9.3.1 ISharedPropertyGroupManager._NewEnum Property

Pascal

```
property _NewEnum: IUnknown;
```

C++

```
__property IUnknown _NewEnum;
```

Description

This is _NewEnum, a member of class ISharedPropertyGroupManager.

42.1.9.3.2 ISharedPropertyGroupManager.Group Property

Pascal

```
property Group [const Name: WideString]: ISharedPropertyGroup;
```

C++

```
__property ISharedPropertyGroup Group[BSTR const Name];
```

Description

This is Group, a member of class ISharedPropertyGroupManager.

42.1.10 ITransactionContext Interface

Class Hierarchy

Pascal

```
ITransactionContext = interface(IDispatch);
```

C++

```
__interface ITransactionContext : public IDispatch;
```

File

Mtx

Description

This is class Mtx.ITransactionContext.

42.1.10.1 ITransactionContext Members

The following tables list the members exposed by ITransactionContext.

Public Methods

	Name	Description
	Abort (see page 2434)	This is Abort, a member of class ITransactionContext.
	Commit (see page 2434)	This is Commit, a member of class ITransactionContext.
	CreateInstance (see page 2435)	This is CreateInstance, a member of class ITransactionContext.

42.1.10.2 ITransactionContext Methods

The methods of the ITransactionContext class are listed here.

Public Methods

	Name	Description
	Abort (see page 2434)	This is Abort, a member of class ITransactionContext.
	Commit (see page 2434)	This is Commit, a member of class ITransactionContext.
	CreateInstance (see page 2435)	This is CreateInstance, a member of class ITransactionContext.

42.1.10.2.1 ITransactionContext.Abort Method

Pascal

```
procedure Abort; safecall;
```

C++

```
__fastcall Abort();
```

Description

This is Abort, a member of class ITransactionContext.

42.1.10.2.2 ITransactionContext.Commit Method

Pascal

```
procedure Commit; safecall;
```

C++

```
__fastcall Commit();
```

Description

This is Commit, a member of class ITransactionContext.

42.1.10.2.3 ITransactionContext.CreateInstance Method

Pascal

```
function CreateInstance(const pszProgId: WideString): OleVariant; safecall;
```

C++

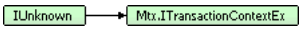
```
__fastcall OleVariant CreateInstance(const BSTR pszProgId);
```

Description

This is CreateInstance, a member of class ITransactionContext.

42.1.11 ITransactionContextEx Interface

Class Hierarchy



Pascal

```
ITransactionContextEx = interface(IUnknown);
```

C++

```
__interface ITransactionContextEx : public IUnknown;
```

File

Mtx

Description

This is class Mtx.ITransactionContextEx.

42.1.11.1 ITransactionContextEx Members

The following tables list the members exposed by ITransactionContextEx.

Public Methods

	Name	Description
	Abort (see page 2436)	This is Abort, a member of class ITransactionContextEx.
	Commit (see page 2436)	This is Commit, a member of class ITransactionContextEx.
	CreateInstance (see page 2436)	This is CreateInstance, a member of class ITransactionContextEx.

42.1.11.2 ITransactionContextEx Methods

The methods of the ITransactionContextEx class are listed here.

Public Methods

	Name	Description
	Abort (see page 2436)	This is Abort, a member of class ITransactionContextEx.
	Commit (see page 2436)	This is Commit, a member of class ITransactionContextEx.
	CreateInstance (see page 2436)	This is CreateInstance, a member of class ITransactionContextEx.

42.1.11.2.1 ITransactionContextEx.Abort Method

Pascal

```
function Abort: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT Abort();
```

Description

This is Abort, a member of class ITransactionContextEx.

42.1.11.2.2 ITransactionContextEx.Commit Method

Pascal

```
function Commit: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT Commit();
```

Description

This is Commit, a member of class ITransactionContextEx.

42.1.11.2.3 ITransactionContextEx.CreateInstance Method

Pascal

```
function CreateInstance(const cid: TGUID; const rid: TGUID; out pv): HRESULT; stdcall;
```

C++

```
__fastcall __stdcall HRESULT CreateInstance(const TGUID cid, const TGUID rid,  pv);
```

Description

This is CreateInstance, a member of class ITransactionContextEx.

42.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	CreateSharedPropertyGroup (see page 2437)	Returns the interface for a SharedPropertyGroup that allows MTS objects within a server process to share state.
	CreateSharedPropertyGroupManager (see page 2438)	Returns an interface for a new shared property group manager that creates and coordinates shared properties of MTS objects.
	CreateTransactionContext (see page 2438)	Starts an MTS transaction.
	CreateTransactionContextEx (see page 2439)	Instantiates a TransactionContextEx object, which marks the beginning of a transaction.
	GetObjectContext (see page 2440)	Fetches the IObjectContext (see page 2420) interface for the current thread.
	SafeRef (see page 2440)	Returns a reference to a transactional object that is safe to pass outside its context.

42.2.1 Mtx.CreateSharedPropertyGroup Function

Returns the interface for a SharedPropertyGroup that allows MTS objects within a server process to share state.

Pascal

```
function CreateSharedPropertyGroup(const Name: WideString): ISharedPropertyGroup;
```

C++

```
ISharedPropertyGroup CreateSharedPropertyGroup(const BSTR Name);
```

File

Mtx

Description

CreateSharedPropertyGroup returns an interface to a new shared property group. If a property group of the specified name exists, it returns a reference to the existing shared property group. Use this function to create and access the shared properties of a shared property group.

CreateSharedPropertyGroup sets up a SharedPropertyGroup with the default values of LockSetGet (see page 2443) for locking (isolation) mode, and Process (see page 2447) for release mode. LockSetGet (see page 2443) locks a property during every get or set operation on a shared property, making each operation atomic. This ensures that two clients cannot read or write to the same property at the same time, but it does not prevent other clients from concurrently accessing other properties in the same group. The Process (see page 2447) release mode ensures that the property group is not destroyed until the process in which it was created has terminated. (Objects that hold references on a property group must still call Release on their references).

To override these defaults, call CreateSharedPropertyGroupManager (see page 2438) and use the resulting interface to create a property group.

ISharedPropertyGroup (see page 2429) has several methods that can be used to share state among multiple objects in a server process. See ISharedPropertyGroup (see page 2429) in the MTS documentation for a list of methods.

By using shared property groups, you eliminate name collisions because they establish unique name spaces for the shared

properties they contain. The Shared Property Manager also implements locks and semaphores to protect shared properties from simultaneous access, which could result in lost updates and could leave the properties in an unpredictable state.

Note: Only objects running in the same process can share properties. If you want instances of different components to share properties, you have to install the components in the same MTS package. Because there is a risk that administrators will move components from one package to another, it's safest to limit the use of a shared property group to instances of components that are defined in the same DLL.

See Also

CreateSharedPropertyGroupManager (see page 2438)

42.2.2 Mtx.CreateSharedPropertyGroupManager Function

Returns an interface for a new shared property group manager that creates and coordinates shared properties of MTS objects.

Pascal

```
function CreateSharedPropertyGroupManager: ISharedPropertyGroupManager;
```

C++

```
ISharedPropertyGroupManager CreateSharedPropertyGroupManager();
```

File

Mtx

Description

Call CreateSharedPropertyGroupManager before creating shared property groups if you are not using the CreateSharedPropertyGroup (see page 2437) function. The ISharedPropertyGroupManager (see page 2431) interface lets you create and access shared properties with more control that is provided by CreateSharedPropertyGroup (see page 2437).

By using shared property groups, you eliminate name collisions because they establish unique name spaces for the shared properties they contain. The Shared Property Manager also implements locks and semaphores to protect shared properties from simultaneous access, which could result in lost updates and could leave the properties in an unpredictable state.

For more information about the ISharedPropertyGroupManager (see page 2431) interface, see the MTS documentation.

See Also

CreateSharedPropertyGroup (see page 2437)

42.2.3 Mtx.CreateTransactionContext Function

Starts an MTS transaction.

Pascal

```
function CreateTransactionContext: ITransactionContext;
```

C++

```
ITransactionContext CreateTransactionContext();
```

File

Mtx

Description

CreateTransactionContext allows client applications to control MTS transactions. CreateTransactionContext returns an interface to the new transaction context. Use this interface to enlist MTS server objects in the transaction, and to commit or roll back the transaction when done.

See Also

CreateTransactionContextEx (see page 2439)

TMtsAutoObject (see page 2388)

TMtsDataModule (see page 2398)

42.2.4 Mtx.CreateTransactionContextEx Function

Instantiates a TransactionContextEx object, which marks the beginning of a transaction.

Pascal

```
function CreateTransactionContextEx: ITransactionContextEx;
```

C++

```
ITransactionContextEx CreateTransactionContextEx();
```

File

Mtx

Description

MTS base clients use CreateTransactionEx to combine the work of one or more MTS objects into an atomic transaction and to commit or abort the transaction. Clients initiate a transaction by instantiating a CreateTransactionEx object. Clients end a transaction by calling Complete or Abort on the object.

Method	Description
Abort	Aborts the work of all MTS objects participating in the current transaction. The transaction is completed on return from this method.
Commit	Attempts to commit the work of all MTS objects participating in the current transaction. If any of the MTS objects participating in the transaction call SetAbort or DisableCommit, or if a system error occurs, the transaction is aborted. Otherwise, the transaction is committed. In either case, the transaction is completed on return from this method.
CreateInstance	Instantiates another MTS object. If the component that provides the object is configured to support or require a transaction, then the new object runs under the transaction of the TransactionContextEx object.

See Also

CreateTransactionContext (see page 2438)

TMtsAutoObject (see page 2388)

TMtsDataModule (see page 2398)

42.2.5 Mtx.GetObjectContext Function

Fetches the IObjectContext (see page 2420) interface for the current thread.

Pascal

```
function GetObjectContext: IObjectContext;
```

C++

```
IObjectContext GetObjectContext();
```

File

Mtx

Description

When a COM object is created as part of the MTS environment or of a COM+ package (that is, a transactional object), a corresponding context object is also created. This context object exists for the entire lifetime of its transactional object, across one or more reactivation cycles. The object context keeps track of the object during deactivation. Call GetObjectContext to obtain an IObjectContext (see page 2420) interface pointer for the context object associated with the current thread.

Pass a pointer that receives the IObjectContext (see page 2420) interface pointer as ppInstanceContext.

GetObjectContext returns S_OK if it returns a pointer to the context object. Otherwise, it returns an error code.

Note: C++Builder automatically fetches the IObjectContext (see page 2420) interface when a transactional object is activated. The interface pointer is stored as the m_spObjectContext member of the transactional object's class.

Warning: When writing a transactional object that uses VCL classes, use the Get_ObjectContext method of TMtsDll instead. ComSvcs.h defines GetObjectContext using a macro so that it calls COM+ or MTS, depending on the system you use to build the transactional object. This macro interferes with the GetObjectContext method of some VCL classes, and so it is undefined when you include the VCL headers.

42.2.6 Mtx.SafeRef Function

Returns a reference to a transactional object that is safe to pass outside its context.

Pascal

```
function SafeRef(const rid: TGUID; Unk: IUnknown): Pointer;
```

C++

```
void * SafeRef(const TGUID rid, IUnknown Unk);
```

File

Mtx

Description

SafeRef encapsulates the SafeRef MTS function that passes in

a reference to the interface ID of the interface (rid) that the current object wants to pass to another object or client

and

a reference to an interface on the current object (pUnk).

SafeRef returns a pointer to the interface specified in the rid parameter that is safe to pass outside the current object's context.

When using COM+, it is always safe to pass a self-reference to a client or other object. Thus, when you call SafeRef in an application that is running under a version of Windows that implemnts COM+, SafeRef simply returns the pointer passed as pUnk. Thus, it is not necessary to call SafeRef for COM+ applications. It will, however, do no harm in an application that is used under both MTS and COM+.

Under MTS, and should always call SafeRef before passing a self-reference to a client or other object and then pass the reference returned by this call. Under MTS, it is not safe for an object to pass a self pointer (Delphi) or this pointer (C++), or a self-reference obtained through an internal call to QueryInterface, to a client or to any other object. Once such a reference is passed outside the object's context, it is no longer a valid reference.

Calling SafeRef on a reference that is already safe returns the safe reference unchanged, except that the reference count on the interface is incremented.

When a client calls QueryInterface on a reference that is safe, MTS automatically ensures that the reference returned to the client is also a safe reference.

An object that obtains a safe reference must release the safe reference when it is finished with it.

Under MTS, safe references have different pointer values than their unsafe counterparts. For example, self or this and the safe version of self or this do not have the same value. It is important to be aware of this when testing whether two pointers refer to the same object. Calling QueryInterface for IID_IUnknown on each of the pointers and comparing the value of the returned pointers may result in the wrong conclusion. It is possible that both pointers refer to the same object, but that one is a safe reference and the other is not. If both references are safe references, they can be compared in the usual way. This is only a consideration for MTS objects, because clients should never have access to unsafe references.

Note: When writing a transactional object that may be installed under either MTS or COM+, use the SafeRef method of TMtsDll instead. TMtsDll automatically checks the current environment and returns the appropriate reference.

See Also

TMtsAutoObject (see page 2388)

42.3 Constants

The following table lists constants in this documentation.

Constants

Name	Description
CLASS_SharedPropertyGroupManager (see page 2442)	This is constant Mtx.CLASS_SharedPropertyGroupManager.
CLASS_TransactionContext (see page 2442)	This is constant Mtx.CLASS_TransactionContext.
CLASS_TransactionContextEx (see page 2443)	This is constant Mtx.CLASS_TransactionContextEx.
LockMethod (see page 2443)	This is constant Mtx.LockMethod.
LockSetGet (see page 2443)	This is constant Mtx.LockSetGet.
mtsErrCtxAborted (see page 2444)	This is constant Mtx.mtsErrCtxAborted.
mtsErrCtxAborting (see page 2444)	This is constant Mtx.mtsErrCtxAborting.
mtsErrCtxActivityTimeout (see page 2444)	This is constant Mtx.mtsErrCtxActivityTimeout.
mtsErrCtxNoContext (see page 2445)	This is constant Mtx.mtsErrCtxNoContext.
mtsErrCtxNoSecurity (see page 2445)	This is constant Mtx.mtsErrCtxNoSecurity.
mtsErrCtxNotRegistered (see page 2445)	This is constant Mtx.mtsErrCtxNotRegistered.
mtsErrCtxOldReference (see page 2446)	This is constant Mtx.mtsErrCtxOldReference.
mtsErrCtxRoleNotFound (see page 2446)	This is constant Mtx.mtsErrCtxRoleNotFound.
mtsErrCtxTMNotAvailable (see page 2446)	This is constant Mtx.mtsErrCtxTMNotAvailable.
mtsErrCtxWrongThread (see page 2447)	This is constant Mtx.mtsErrCtxWrongThread.
Process (see page 2447)	This is constant Mtx.Process.
Standard (see page 2447)	This is constant Mtx.Standard.

42.3.1 Mtx.CLASS_SharedPropertyGroupManager Constant

Pascal

```
CLASS_SharedPropertyGroupManager: TGUID = '{2A005C11-A5DE-11CF-9E66-00AA00A3F464}';
```

C++

```
TGUID CLASS_SharedPropertyGroupManager = '{2A005C11-A5DE-11CF-9E66-00AA00A3F464}';
```

File

Mtx

Description

This is constant Mtx.CLASS_SharedPropertyGroupManager.

42.3.2 Mtx.CLASS_TransactionContext Constant

Pascal

```
CLASS_TransactionContext: TGUID = '{7999FC25-D3C6-11CF-ACAB-00A024A55AEF}';
```

C++

```
TGUID CLASS_TransactionContext = '{7999FC25-D3C6-11CF-ACAB-00A024A55AEF}';
```

File

Mtx

Description

This is constant Mtx.CLASS_TransactionContext.

42.3.3 Mtx.CLASS_TransactionContextEx Constant

Pascal

```
CLASS_TransactionContextEx: TGUID = '{5CB66670-D3D4-11CF-ACAB-00A024A55AEF}';
```

C++

```
TGUID CLASS_TransactionContextEx = '{5CB66670-D3D4-11CF-ACAB-00A024A55AEF}';
```

File

Mtx

Description

This is constant Mtx.CLASS_TransactionContextEx.

42.3.4 Mtx.LockMethod Constant

Pascal

```
LockMethod = $00000001;
```

C++

```
LockMethod = 0x00000001;
```

File

Mtx

Description

This is constant Mtx.LockMethod.

42.3.5 Mtx.LockSetGet Constant

Pascal

```
LockSetGet = $00000000;
```

C++

```
LockSetGet = 0x00000000;
```

File

Mtx

Description

This is constant Mtx.LockSetGet.

42.3.6 Mtx.mtsErrCtxAborted Constant

Pascal

```
mtsErrCtxAborted = $8004E002;
```

C++

```
mtsErrCtxAborted = 0x8004E002;
```

File

Mtx

Description

This is constant Mtx.mtsErrCtxAborted.

42.3.7 Mtx.mtsErrCtxAborting Constant

Pascal

```
mtsErrCtxAborting = $8004E003;
```

C++

```
mtsErrCtxAborting = 0x8004E003;
```

File

Mtx

Description

This is constant Mtx.mtsErrCtxAborting.

42.3.8 Mtx.mtsErrCtxActivityTimeout Constant

Pascal

```
mtsErrCtxActivityTimeout = $8004E006;
```

C++

```
mtsErrCtxActivityTimeout = 0x8004E006;
```

File

Mtx

Description

This is constant Mtx.mtsErrCtxActivityTimeout.

42.3.9 Mtx.mtsErrCtxNoContext Constant

Pascal

```
mtsErrCtxNoContext = $8004E004;
```

C++

```
mtsErrCtxNoContext = 0x8004E004;
```

File

Mtx

Description

This is constant Mtx.mtsErrCtxNoContext.

42.3.10 Mtx.mtsErrCtxNoSecurity Constant

Pascal

```
mtsErrCtxNoSecurity = $8004E00D;
```

C++

```
mtsErrCtxNoSecurity = 0x8004E00D;
```

File

Mtx

Description

This is constant Mtx.mtsErrCtxNoSecurity.

42.3.11 Mtx.mtsErrCtxNotRegistered Constant

Pascal

```
mtsErrCtxNotRegistered = $8004E005;
```

C++

```
mtsErrCtxNotRegistered = 0x8004E005;
```

File

Mtx

Description

This is constant Mtx.mtsErrCtxNotRegistered.

42.3.12 Mtx.mtsErrCtxOldReference Constant

Pascal

```
mtsErrCtxOldReference = $8004E007;
```

C++

```
mtsErrCtxOldReference = 0x8004E007;
```

File

Mtx

Description

This is constant Mtx.mtsErrCtxOldReference.

42.3.13 Mtx.mtsErrCtxRoleNotFound Constant

Pascal

```
mtsErrCtxRoleNotFound = $8004E00C;
```

C++

```
mtsErrCtxRoleNotFound = 0x8004E00C;
```

File

Mtx

Description

This is constant Mtx.mtsErrCtxRoleNotFound.

42.3.14 Mtx.mtsErrCtxTMNotAvailable Constant

Pascal

```
mtsErrCtxTMNotAvailable = $8004E00F;
```

C++

```
mtsErrCtxTMNotAvailable = 0x8004E00F;
```

File

Mtx

Description

This is constant Mtx.mtsErrCtxTMNotAvailable.

42.3.15 Mtx.mtsErrCtxWrongThread Constant

Pascal

```
mtsErrCtxWrongThread = $8004E00E;
```

C++

```
mtsErrCtxWrongThread = 0x8004E00E;
```

File

Mtx

Description

This is constant Mtx.mtsErrCtxWrongThread.

42.3.16 Mtx.Process Constant

Pascal

```
Process = $00000001;
```

C++

```
Process = 0x00000001;
```

File

Mtx

Description

This is constant Mtx.Process.

42.3.17 Mtx.Standard Constant

Pascal

```
Standard = $00000000;
```

C++

```
Standard = 0x00000000;
```

File

Mtx

Description

This is constant Mtx.Standard.

43 ObjAuto Namespace

This is namespace ObjAuto.

Topics

Name	Description
Interfaces (see page 2451)	The following table lists interfaces in this documentation.
Functions (see page 2453)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 2456)	The following table lists structs, records, enums in this documentation.
Types (see page 2459)	The following table lists types in this documentation.
Constants (see page 2463)	The following table lists constants in this documentation.

Constants

Name	Description
paEAX (see page 2463)	This is constant ObjAuto.paEAX.
paECX (see page 2463)	This is constant ObjAuto.paECX.
paEDX (see page 2464)	This is constant ObjAuto.paEDX.
paStack (see page 2464)	This is constant ObjAuto.paStack.

Functions

	Name	Description
	CreateMethodPointer (see page 2453)	This is function ObjAuto.CreateMethodPointer.
	CreateMethodPointer (see page 2453)	This is function ObjAuto.CreateMethodPointer.
	GetInvokeInstance (see page 2454)	This is function ObjAuto.GetInvokeInstance.
	GetMethodInfo (see page 2454)	This is function ObjAuto.GetMethodInfo.
	GetMethods (see page 2454)	This is function ObjAuto.GetMethods.
	ObjectInvoke (see page 2455)	This is function ObjAuto.ObjectInvoke.
	ReleaseMethodPointer (see page 2455)	This is function ObjAuto.ReleaseMethodPointer.

Interfaces

	Name	Description
	IMethodHandler (see page 2451)	This is class ObjAuto.IMethodHandler.

Structs, Records, Enums

	Name	Description
	TCallingConvention (see page 2456)	This is record ObjAuto.TCallingConvention.
	TMethodInfoHeader (see page 2456)	This is record ObjAuto.TMethodInfoHeader.

	TParameters (see page 2457)	This is record ObjAuto.TParameters.
	TParamInfo (see page 2457)	This is record ObjAuto.TParamInfo.
	TReturnInfo (see page 2458)	This is record ObjAuto.TReturnInfo.

Types

Name	Description
PMethodInfoHeader (see page 2459)	This is type ObjAuto.PMethodInfoHeader.
PParameters (see page 2459)	This is type ObjAuto.PParameters.
PParamInfo (see page 2460)	This is type ObjAuto.PParamInfo.
PPointer (see page 2460)	This is type ObjAuto.PPointer.
PReturnInfo (see page 2460)	This is type ObjAuto.PReturnInfo.
PWord (see page 2461)	This is type ObjAuto.PWord.
TDynamicInvokeEvent (see page 2461)	This is type ObjAuto.TDynamicInvokeEvent.
TMethodInfoArray (see page 2461)	This is type ObjAuto.TMethodInfoArray.
TParamFlags (see page 2462)	This is type ObjAuto.TParamFlags.

43.1 Interfaces

The following table lists interfaces in this documentation.

Interfaces

	Name	Description
	IMethodHandler (see page 2451)	This is class ObjAuto.IMethodHandler.

43.1.1 IMethodHandler Interface

Class Hierarchy

ObjAuto.IMethodHandler

Pascal

```
IMethodHandler = interface;
```

C++

```
__interface IMethodHandler;
```

File

ObjAuto

Description

This is class ObjAuto.IMethodHandler.

43.1.1.1 IMethodHandler Members

The following tables list the members exposed by IMethodHandler.

Public Methods

	Name	Description
	Execute (see page 2452)	This is Execute, a member of class IMethodHandler.
	InstanceToVariant (see page 2452)	This is InstanceToVariant, a member of class IMethodHandler.

43.1.1.2 IMethodHandler Methods

The methods of the IMethodHandler class are listed here.

Public Methods

	Name	Description
	Execute (see page 2452)	This is Execute, a member of class IMethodHandler.
	InstanceToVariant (see page 2452)	This is InstanceToVariant, a member of class IMethodHandler.

43.1.1.2.1 IMethodHandler.Execute Method

Pascal

```
function Execute(const Args: array of Variant): Variant;
```

C++

```
__fastcall Variant Execute(const array of Variant Args);
```

Description

This is Execute, a member of class IMethodHandler.

43.1.1.2.2 IMethodHandler.InstanceToVariant Method

Pascal

```
function InstanceToVariant(Instance: TObject): Variant;
```

C++

```
__fastcall Variant InstanceToVariant(TObject * Instance);
```

Description

This is InstanceToVariant, a member of class IMethodHandler.

43.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	CreateMethodPointer (see page 2453)	This is function ObjAuto.CreateMethodPointer.
	CreateMethodPointer (see page 2453)	This is function ObjAuto.CreateMethodPointer.
	GetInvokeInstance (see page 2454)	This is function ObjAuto.GetInvokeInstance.
	GetMethodInfo (see page 2454)	This is function ObjAuto.GetMethodInfo.
	GetMethods (see page 2454)	This is function ObjAuto.GetMethods.
	ObjectInvoke (see page 2455)	This is function ObjAuto.ObjectInvoke.
	ReleaseMethodPointer (see page 2455)	This is function ObjAuto.ReleaseMethodPointer.

43.2.1 ObjAuto.CreateMethodPointer Function

Pascal

```
function CreateMethodPointer(const MethodHandler: IMethodHandler; TypeData: PTypeData):
TMethod; overload;
```

C++

```
TMethod CreateMethodPointer(const IMethodHandler MethodHandler, PTypeData TypeData);
```

File

ObjAuto

Description

This is function ObjAuto.CreateMethodPointer.

43.2.2 ObjAuto.CreateMethodPointer Function

Pascal

```
function CreateMethodPointer(const ADynamicInvokeEvent: TDynamicInvokeEvent; TypeData:
PTypeData): TMethod; overload;
```

C++

```
TMethod CreateMethodPointer(const TDynamicInvokeEvent ADynamicInvokeEvent, PTypeData TypeData);
```

File

ObjAuto

Description

This is function ObjAuto.CreateMethodPointer.

43.2.3 ObjAuto.GetInvokeInstance Function

Pascal

```
function GetInvokeInstance(MethodPointer: TMethod): TObject;
```

C++

```
TObject * GetInvokeInstance(TMethod MethodPointer);
```

File

ObjAuto

Description

This is function ObjAuto.GetInvokeInstance.

43.2.4 ObjAuto.GetMethodInfo Function

Pascal

```
function GetMethodInfo(Instance: TObject; const MethodName: ShortString): PMethodInfoHeader;
```

C++

```
PMethodInfoHeader GetMethodInfo(TObject * Instance, const ShortString MethodName);
```

File

ObjAuto

Description

This is function ObjAuto.GetMethodInfo.

43.2.5 ObjAuto.GetMethods Function

Pascal

```
function GetMethods(ClassType: TClass): TMethodInfoArray;
```

C++

```
TMethodInfoArray GetMethods(TClass ClassType);
```

File

ObjAuto

Description

This is function ObjAuto.GetMethods.

43.2.6 ObjAuto.ObjectInvoke Function

Pascal

```
function ObjectInvoke(Instance: TObject; MethodHeader: PMethodInfoHeader; const ParamIndexes:
array of Integer; const Params: array of Variant): Variant;
```

C++

```
Variant ObjectInvoke(TObject * Instance, PMethodInfoHeader MethodHeader, const array of
Integer ParamIndexes, const array of Variant Params);
```

File

ObjAuto

Description

This is function ObjAuto.ObjectInvoke.

43.2.7 ObjAuto.ReleaseMethodPointer Function

Pascal

```
procedure ReleaseMethodPointer(MethodPointer: TMethod);
```

C++

```
ReleaseMethodPointer(TMethod MethodPointer);
```

File

ObjAuto

Description

This is function ObjAuto.ReleaseMethodPointer.

43.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TCallingConvention (see page 2456)	This is record ObjAuto.TCallingConvention.

Records

	Name	Description
	TMethodInfoHeader (see page 2456)	This is record ObjAuto.TMethodInfoHeader.
	TParameters (see page 2457)	This is record ObjAuto.TParameters.
	TParamInfo (see page 2457)	This is record ObjAuto.TParamInfo.
	TReturnInfo (see page 2458)	This is record ObjAuto.TReturnInfo.

43.3.1 ObjAuto.TCallingConvention Enumeration

Pascal

```
TCallingConvention = (
  ccRegister,
  ccCdecl,
  ccPascal,
  ccStdCall,
  ccSafeCall
);
```

C++

```
enum TCallingConvention {
  ccRegister,
  ccCdecl,
  ccPascal,
  ccStdCall,
  ccSafeCall
};
```

File

ObjAuto

Description

This is record ObjAuto.TCallingConvention.

43.3.2 ObjAuto.TMethodInfoHeader Record

Pascal

```
TMethodInfoHeader = packed record
  Len: Word;
  Addr: Pointer;
```

```
    Name: ShortString;
  end;
```

C++

```
  struct TMethodInfoHeader {
    Word Len;
    void * Addr;
    ShortString Name;
  };
```

File

ObjAuto

Description

This is record ObjAuto.TMethodInfoHeader.

43.3.3 ObjAuto.TParameters Record

Pascal

```
  TParameters = packed record
    Registers: array[paEDX..paECX] of Cardinal;
    Stack: array[0..1023] of Byte;
  end;
```

C++

```
  struct TParameters {
    array[paEDX..paECX] of Cardinal Registers;
    array[0..1023] of Byte Stack;
  };
```

File

ObjAuto

Description

This is record ObjAuto.TParameters.

43.3.4 ObjAuto.TParamInfo Record

Pascal

```
  TParamInfo = packed record
    Flags: TParamFlags;
    ParamType: ^PTypeInfo;
    Access: Word;
    Name: ShortString;
  end;
```

C++

```
  struct TParamInfo {
    TParamFlags Flags;
    ^PTypeInfo ParamType;
    Word Access;
    ShortString Name;
  };
```

File

ObjAuto

Description

This is record ObjAuto.TParamInfo.

43.3.5 ObjAuto.TReturnInfo Record

Pascal

```
TReturnInfo = packed record
  Version: Byte;
  CallingConvention: TCallingConvention;
  ReturnType: ^PTypeInfo;
  ParamSize: Word;
end;
```

C++

```
struct TReturnInfo {
  Byte Version;
  TCallingConvention CallingConvention;
  ^PTypeInfo ReturnType;
  Word ParamSize;
};
```

File

ObjAuto

Description

This is record ObjAuto.TReturnInfo.

43.4 Types

The following table lists types in this documentation.

Types

Name	Description
PMethodInfoHeader (see page 2459)	This is type ObjAuto.PMethodInfoHeader.
PParameters (see page 2459)	This is type ObjAuto.PParameters.
PParamInfo (see page 2460)	This is type ObjAuto.PParamInfo.
PPointer (see page 2460)	This is type ObjAuto.PPointer.
PReturnInfo (see page 2460)	This is type ObjAuto.PReturnInfo.
PWord (see page 2461)	This is type ObjAuto.PWord.
TDynamicInvokeEvent (see page 2461)	This is type ObjAuto.TDynamicInvokeEvent.
TMethodInfoArray (see page 2461)	This is type ObjAuto.TMethodInfoArray.
TParamFlags (see page 2462)	This is type ObjAuto.TParamFlags.

43.4.1 ObjAuto.PMethodInfoHeader Type

Pascal

```
PMethodInfoHeader = ^TMethodInfoHeader;
```

C++

```
^TMethodInfoHeader PMethodInfoHeader;
```

File

ObjAuto

Description

This is type ObjAuto.PMethodInfoHeader.

43.4.2 ObjAuto.PParameters Type

Pascal

```
PParameters = ^TParameters;
```

C++

```
^TParameters PParameters;
```

File

ObjAuto

Description

This is type ObjAuto.PParameters.

43.4.3 ObjAuto.PParamInfo Type

Pascal

```
PParamInfo = ^TParamInfo;
```

C++

```
^TParamInfo PParamInfo;
```

File

ObjAuto

Description

This is type ObjAuto.PParamInfo.

43.4.4 ObjAuto.PPointer Type

Pascal

```
PPointer = ^Pointer;
```

C++

```
^Pointer PPointer;
```

File

ObjAuto

Description

This is type ObjAuto.PPointer.

43.4.5 ObjAuto.PReturnInfo Type

Pascal

```
PReturnInfo = ^TReturnInfo;
```

C++

```
^TReturnInfo PReturnInfo;
```

File

ObjAuto

Description

This is type ObjAuto.PReturnInfo.

43.4.6 ObjAuto.PWord Type

Pascal

```
PWord = ^Word;
```

C++

```
^Word PWord;
```

File

ObjAuto

Description

This is type ObjAuto.PWord.

43.4.7 ObjAuto.TDynamicInvokeEvent Type

Pascal

```
TDynamicInvokeEvent = procedure (Params: PParameters; StackSize: Integer) of object;
```

C++

```
(Params: PParameters; StackSize: Integer) ( TDynamicInvokeEvent)();
```

File

ObjAuto

Description

This is type ObjAuto.TDynamicInvokeEvent.

43.4.8 ObjAuto.TMethodInfoArray Type

Pascal

```
TMethodInfoArray = array of PMethodInfoHeader;
```

C++

```
array of PMethodInfoHeader TMethodInfoArray;
```

File

ObjAuto

Description

This is type ObjAuto.TMethodInfoArray.

43.4.9 ObjAuto.TParamFlags Type

Pascal

```
TParamFlags = set of (pfVar, pfConst, pfArray, pfAddress, pfReference, pfOut, pfResult);
```

C++

```
(pfVar, pfConst, pfArray, pfAddress, pfReference, pfOut, pfResult) TParamFlags;
```

File

ObjAuto

Description

This is type ObjAuto.TParamFlags.

43.5 Constants

The following table lists constants in this documentation.

Constants

Name	Description
paEAX (see page 2463)	This is constant ObjAuto.paEAX.
paECX (see page 2463)	This is constant ObjAuto.paECX.
paEDX (see page 2464)	This is constant ObjAuto.paEDX.
paStack (see page 2464)	This is constant ObjAuto.paStack.

43.5.1 ObjAuto.paEAX Constant

Pascal

```
paEAX = Word(0);
```

C++

```
paEAX = Word(0);
```

File

ObjAuto

Description

This is constant ObjAuto.paEAX.

43.5.2 ObjAuto.paECX Constant

Pascal

```
paECX = Word(2);
```

C++

```
paECX = Word(2);
```

File

ObjAuto

Description

This is constant ObjAuto.paECX.

43.5.3 ObjAuto.paEDX Constant

Pascal

```
paEDX = Word(1);
```

C++

```
paEDX = Word(1);
```

File

ObjAuto

Description

This is constant ObjAuto.paEDX.

43.5.4 ObjAuto.paStack Constant

Pascal

```
paStack = Word(3);
```

C++

```
paStack = Word(3);
```

File

ObjAuto

Description

This is constant ObjAuto.paStack.

44 ObjBrkr Namespace

This is namespace ObjBrkr.

Topics

Name	Description
Classes (see page 2466)	The following table lists classes in this documentation.

Classes

	Name	Description
	EBrokerException (see page 2466)	EBrokerException is the exception raised by simple object brokers when they can't locate an available server.
	TServerCollection (see page 2466)	TServerCollection is a collection of TServerItem (see page 2470) objects.
	TServerItem (see page 2470)	TServerItem describes an application server for a simple object broker.
	TSimpleObjectBroker (see page 2475)	TSimpleObjectBroker maintains a list of available application servers for a connection component.

44.1 Classes

The following table lists classes in this documentation.

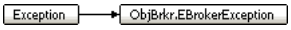
Classes

	Name	Description
	EBrokerException (see page 2466)	EBrokerException is the exception raised by simple object brokers when they can't locate an available server.
	TServerCollection (see page 2466)	TServerCollection is a collection of TServerItem (see page 2470) objects.
	TServerItem (see page 2470)	TServerItem describes an application server for a simple object broker.
	TSimpleObjectBroker (see page 2475)	TSimpleObjectBroker maintains a list of available application servers for a connection component.

44.1.1 EBrokerException Class

EBrokerException is the exception raised by simple object brokers when they can't locate an available server.

Class Hierarchy



Pascal

```
EBrokerException = class(Exception);
```

C++

```
class EBrokerException : public Exception;
```

File

ObjBrkr

Description

EBrokerException is raised when a simple object broker can't locate an available server.

See Also

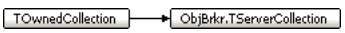
TSimpleObjectBroker (see page 2475)

TServerCollection (see page 2466)

44.1.2 TServerCollection Class

TServerCollection is a collection of TServerItem (see page 2470) objects.

Class Hierarchy



Pascal

```
TServerCollection = class(TOwnedCollection);
```

C++

```
class TServerCollection : public TOwnedCollection;
```

File

ObjBrkr

Description

TSimpleObjectBroker (see page 2475) uses TServerCollection to represent the set of available application servers to which a client application can connect. Use the properties and methods of TServerCollection to manage the set of servers. For example, you can add items as new server systems come on line, disable server items when the corresponding system is temporarily out of commission, and so on.

See Also

TServerItem (see page 2470)

TSimpleObjectBroker (see page 2475)

44.1.2.1 TServerCollection Members

The following tables list the members exposed by TServerCollection.

Public Methods

	Name	Description
	Create (see page 2467)	Creates and initializes a TServerCollection (see page 2466) object.
	FindServer (see page 2468)	Returns the server from the Items (see page 2470) property array that has a specified computer name.
	GetBalancedName (see page 2468)	Returns the ComputerName property of a randomly chosen server from the Items (see page 2470) property array.
	GetNextName (see page 2469)	Returns the ComputerName property of the first available server in the Items (see page 2470) property array.

Public Properties

	Name	Description
	Items (see page 2470)	Lists the TServerItem (see page 2470) objects in the collection.

44.1.2.2 TServerCollection.Create Constructor

Creates and initializes a TServerCollection (see page 2466) object.

Pascal

```
constructor Create(AOwner: TComponent);
```

C++

```
__fastcall TServerCollection(TComponent * AOwner);
```

Description

Applications should not need to instantiate TServerCollection (see page 2466). The simple object broker that uses TServerCollection (see page 2466) to manage its list of available servers creates this object from its own constructor.

See Also

Servers (see page 2481)

44.1.2.3 TServerCollection Methods

The methods of the TServerCollection class are listed here.

Public Methods

	Name	Description
	FindServer (see page 2468)	Returns the server from the Items (see page 2470) property array that has a specified computer name.
	GetBalancedName (see page 2468)	Returns the ComputerName property of a randomly chosen server from the Items (see page 2470) property array.
	GetNextName (see page 2469)	Returns the ComputerName property of the first available server in the Items (see page 2470) property array.

44.1.2.3.1 TServerCollection.FindServer Method

Returns the server from the Items (see page 2470) property array that has a specified computer name.

Pascal

```
function FindServer(const ComputerName: string): TServerItem;
```

C++

```
__fastcall TServerItem FindServer(const UnicodeString ComputerName);
```

Description

Call FindServer to locate a server in the items property array given its ComputerName property. The ComputerName parameter specifies this computer name. Usually, this value is obtained by a previous call to GetBalancedName (see page 2468) or GetNextName (see page 2469).

Note: FindServer returns the first item it finds with the given ComputerName property. Unlike the GetBalancedName (see page 2468) and GetNextName (see page 2469) methods, it does not ignore servers that have failed or been disabled (HasFailed is true, or Enabled is false).

See Also

GetBalancedName (see page 2468)

GetNextName (see page 2469)

ComputerName (see page 2472)

Items (see page 2470)

Enabled (see page 2473)

HasFailed (see page 2474)

44.1.2.3.2 TServerCollection.GetBalancedName Method

Returns the ComputerName property of a randomly chosen server from the Items (see page 2470) property array.

Pascal

```
function GetBalancedName: string;
```

C++

```
__fastcall UnicodeString GetBalancedName();
```

Description

Call GetBalancedName to get the name of an available server. Servers in the Items (see page 2470) property array are considered available when no connection attempts have failed (HasFailed is false) and their Enabled property has not been set to false. GetBalancedName selects an available TServerItem (see page 2470) at random from the Items (see page 2470) property array and returns its ComputerName property.

See Also

Items (see page 2470)

LoadBalanced (see page 2480)

ComputerName (see page 2472)

HasFailed (see page 2474)

Enabled (see page 2473)

GetNextName (see page 2469)

44.1.2.3.3 TServerCollection.GetNextName Method

Returns the ComputerName property of the first available server in the Items (see page 2470) property array.

Pascal

```
function GetNextName: string;
```

C++

```
__fastcall UnicodeString GetNextName();
```

Description

Call GetNextName to get the name of an available server. Servers in the Items (see page 2470) property array are considered available when no connection attempts have failed (HasFailed is false) and their Enabled property has not been set to false. GetNextName selects the first available TServerItem (see page 2470) in the Items (see page 2470) property array and returns its ComputerName property.

Note: GetNextName always returns the same name as long as the servers' HasFailed and Enabled properties do not change. To choose an available server at random, use GetBalancedName (see page 2468) instead.

See Also

Items (see page 2470)

ComputerName (see page 2472)

HasFailed (see page 2474)

Enabled (see page 2473)

GetBalancedName (see page 2468)

44.1.2.4 TServerCollection Properties

The properties of the TServerCollection class are listed here.

Public Properties

	Name	Description
	Items (see page 2470)	Lists the TServerItem (see page 2470) objects in the collection.

44.1.2.4.1 TServerCollection.Items Property

Lists the TServerItem (see page 2470) objects in the collection.

Pascal

```
property Items [Index: Integer]: TServerItem;
```

C++

```
__property TServerItem Items[int Index];
```

Description

Use Items to access individual TServerItem (see page 2470) objects maintained by TServerCollection (see page 2466).

Note: In Delphi, Items is the default property of TServerCollection (see page 2466), so the name of the Items property can be left off when using this property. Thus the line

```
TSimpleObjectBroker1.Servers.Items[I].Enabled := False;
```

can be written

```
TSimpleObjectBroker1.Servers[I].Enabled := False;
```

Note: In C++, Items can be accessed using the [] operator, to achieve an affect similar to the default property in Delphi.

See Also

Enabled (see page 2473)

GetBalancedName (see page 2468)

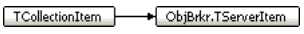
GetNextName (see page 2469)

FindServer (see page 2468)

44.1.3 TServerItem Class

TServerItem describes an application server for a simple object broker.

Class Hierarchy



Pascal

```
TServerItem = class(TCollectionItem);
```

C++

```
class TServerItem : public TCollectionItem;
```

File

ObjBrkr

Description

Use TServerItem to describe an application server to which client applications can connect. TSimpleObjectBroker (see page 2475) maintains a list of these descriptions. Then, when a connection component needs to open a connection, it can use the information stored in a server item to identify an available application server.

See Also

TSimpleObjectBroker (see page 2475)

TRemoteDataModule

TDispatchConnection (see page 1290)

44.1.3.1 TServerItem Members

The following tables list the members exposed by TServerItem.

Protected Methods

	Name	Description
	GetDisplayName (see page 2472)	Returns the name of the collection item as it appears in the collection editor.

Public Methods

	Name	Description
	Create (see page 2471)	Creates and initializes a TServerItem (see page 2470) instance.

Public Properties

	Name	Description
	HasFailed (see page 2474)	Indicates whether a connection attempt to the associated server machine has failed.

Published Properties

	Name	Description
	ComputerName (see page 2472)	Identifies the machine where the application server resides.
	Enabled (see page 2473)	Enables or disables the server item.
	Port (see page 2474)	Specifies the port number for the associated server application.

44.1.3.2 TServerItem.Create Constructor

Creates and initializes a TServerItem (see page 2470) instance.

Pascal

```
constructor Create(AOwner: TCollection); override;
```

C++

```
virtual __fastcall TServerItem(TCollection AOwner);
```

Description

Do not call Create directly to add a new server item to a server collection. Instead, use the Add or Insert method of the TServerCollection (see page 2466) object.

The AOwner parameter specifies the server collection that manages the newly created server item.

After calling the inherited constructor, Create initializes the Enabled (see page 2473) property to true, the HasFailed (see page 2474) property to false, and the Port (see page 2474) property to 211.

See Also

Add

Insert

Port (see page 2474)

HasFailed (see page 2474)

Enabled (see page 2473)

44.1.3.3 TServerItem Methods

The methods of the TServerItem class are listed here.

Protected Methods

	Name	Description
	GetDisplayName (see page 2472)	Returns the name of the collection item as it appears in the collection editor.

44.1.3.3.1 TServerItem.GetDisplayName Method

Returns the name of the collection item as it appears in the collection editor.

Pascal

```
function GetDisplayName: string; override;
```

C++

```
virtual __fastcall UnicodeString GetDisplayName();
```

Description

GetDisplayName is the protected read implementation of the DisplayName property.

See Also

DisplayName

GetNamePath

44.1.3.4 TServerItem Properties

The properties of the TServerItem class are listed here.

Public Properties

	Name	Description
	HasFailed (see page 2474)	Indicates whether a connection attempt to the associated server machine has failed.

Published Properties

	Name	Description
	ComputerName (see page 2472)	Identifies the machine where the application server resides.
	Enabled (see page 2473)	Enables or disables the server item.
	Port (see page 2474)	Specifies the port number for the associated server application.

44.1.3.4.1 TServerItem.ComputerName Property

Identifies the machine where the application server resides.

Pascal

```
property ComputerName: string;
```

C++

```
__property UnicodeString ComputerName;
```

Description

Set ComputerName to a string that identifies the machine where the application server resides. The value of this string depends on the type of connection component that requests this value from its object broker.

If the connection component is a TDCOMConnection (see page 1284) object, ComputerName is the Windows computer name. (This value can be found in the Control panel under network settings.)

If the connection component is a TSocketConnection object, ComputerName is either the host name or IP address of the system running the application server. Socket connection components parse this string to determine which it is.

If the connection component is a TWebConnection or TSoapConnection object, ComputerName is the URL of the dispatcher (httpsrvr.dll or soapsrvr.dll (running on the system with the application server.

See Also

GetBalancedName (see page 2468)

GetNextName (see page 2469)

Port (see page 2474)

ComputerName (see page 1287)

Address

Host

ObjectBroker (see page 1300)

44.1.3.4.2 TServerItem.Enabled Property

Enables or disables the server item.

Pascal

```
property Enabled: Boolean;
```

C++

```
__property Boolean Enabled;
```

Description

Set Enabled to indicate whether the server should made available in the simple object broker. When Enabled is true, the ComputerName (see page 2472) property can be supplied to connection components by the object broker's GetComputerForGUID or GetComputerForProgID methods. When Enabled is false, the server item is not returned.

Note: Even when the server item is disabled, it is still returned by TServerCollection (see page 2466)'s FindServer method.

See Also

GetComputerForGUID (see page 2477)

GetComputerForProgID (see page 2477)

HasFailed (see page 2474)

FindServer (see page 2468)

44.1.3.4.3 TServerItem.HasFailed Property

Indicates whether a connection attempt to the associated server machine has failed.

Pascal

```
property HasFailed: Boolean;
```

C++

```
__property Boolean HasFailed;
```

Description

When connection components can't successfully open a connection to an application server, they inform the object broker by calling SetConnectStatus. The simple object broker, in turn, records this fact by setting the HasFailed property to true.

When HasFailed is false, the ComputerName (see page 2472) property can be supplied to connection components by the object broker's GetComputerForGUID or GetComputerForProgID methods. When HasFailed is true, the server item is not returned.

Once a server has failed it is presumed to be down until the application explicitly resets the HasFailed property to false.

Do not use HasFailed to disable server items. Instead, use the Enabled (see page 2473) property to disable servers for any reason other than a failed connection.

Note: Even when HasFailed is true, the server item is still returned by TServerCollection (see page 2466)'s FindServer method.

See Also

SetConnectStatus (see page 2479)

Enabled (see page 2473)

GetComputerForGUID (see page 2477)

GetComputerForProgID (see page 2477)

FindServer (see page 2468)

44.1.3.4.4 TServerItem.Port Property

Specifies the port number for the associated server application.

Pascal

```
property Port: Integer;
```

C++

```
__property int Port;
```

Description

Read Port to obtain a port number when using the server item to supply information for connecting using TCP/IP. This value is returned by the GetPortForComputer method of the simple object broker.

By default, the value of Port is 211, which is the default port number for the socket dispatcher application ScktSrvr.exe. If the server listens on a different port, change this property to indicate the listening port.

Note: Server items that supply a Port number should specify the server system's IP address or host name as the value of ComputerName (see page 2472).

See Also

ComputerName (see page 2472)

44.1.4 TSimpleObjectBroker Class

TSimpleObjectBroker maintains a list of available application servers for a connection component.

Class Hierarchy

Pascal

```
TSimpleObjectBroker = class(TCustomObjectBroker);
```

C++

```
class TSimpleObjectBroker : public TCustomObjectBroker;
```

File

ObjBrkr

Description

TSimpleObjectBroker maintains a list of servers, all of which run the application server for a multi-tiered application. When a connection component requests a server, the simple object broker passes the name of one of the available servers in the list. This process allows client applications to locate server systems dynamically at runtime. If there are redundant server systems, this allows client applications to continue functioning when a server goes down.

TSimpleObjectBroker can randomize the selection of servers from among the currently available servers in its list to provide a simple system of load balancing.

See Also

TDispatchConnection (see page 1290)

TServerCollection (see page 2466)

44.1.4.1 TSimpleObjectBroker Members

The following tables list the members exposed by TSimpleObjectBroker.

Public Methods

	Name	Description
	Create (see page 2476)	Creates and initializes a simple object broker.
	Destroy (see page 2476)	Destroys an instance of TSimpleObjectBroker (see page 2475).
	GetComputerForGUID (see page 2477)	Returns a string that identifies an available server machine.
	GetComputerForProgID (see page 2477)	Returns a string that identifies an available server machine.
	GetPortForComputer (see page 2478)	Returns a port number for connecting to a specified remote server.
	LoadFromStream (see page 2479)	Sets the value of the Servers (see page 2481) property from a stream.
	SaveToStream (see page 2479)	Saves the value of the Servers (see page 2481) property to a stream.
	SetConnectStatus (see page 2479)	Records whether a connection was established to a specified server.

Public Properties

	Name	Description
	ServerData (see page 2481)	Encapsulates the list of servers as an OleVariant.

Published Properties

	Name	Description
	LoadBalanced (see page 2480)	Specifies how the simple object broker picks a server from its list.
	Servers (see page 2481)	Maintains the list of servers that can be provided to connection components.

44.1.4.2 TSimpleObjectBroker.Create Constructor

Creates and initializes a simple object broker.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TSimpleObjectBroker(TComponent * AOwner);
```

Description

Call Create to instantiate a simple object broker at runtime.

After calling the inherited constructor, Create initializes the LoadBalanced (see page 2480) property to false and creates the TServerCollection (see page 2466) that implements the Servers (see page 2481) property.

See Also

Create

Servers (see page 2481)

LoadBalanced (see page 2480)

44.1.4.3 TSimpleObjectBroker.Destroy Destructor

Destroys an instance of TSimpleObjectBroker (see page 2475).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TSimpleObjectBroker();
```

Description

Do not call Destroy directly. Instead, call Free, which verifies that the simple object broker is not nil, and only then calls Destroy.

See Also

System::TObject::Free

44.1.4.4 TSimpleObjectBroker Methods

The methods of the TSimpleObjectBroker class are listed here.

Public Methods

	Name	Description
	GetComputerForGUID (see page 2477)	Returns a string that identifies an available server machine.
	GetComputerForProgID (see page 2477)	Returns a string that identifies an available server machine.
	GetPortForComputer (see page 2478)	Returns a port number for connecting to a specified remote server.
	LoadFromStream (see page 2479)	Sets the value of the Servers (see page 2481) property from a stream.
	SaveToStream (see page 2479)	Saves the value of the Servers (see page 2481) property to a stream.
	SetConnectStatus (see page 2479)	Records whether a connection was established to a specified server.

44.1.4.4.1 TSimpleObjectBroker.GetComputerForGUID Method

Returns a string that identifies an available server machine.

Pascal

```
function GetComputerForGUID(GUID: TGUID): string; override;
```

C++

```
virtual __fastcall UnicodeString GetComputerForGUID(TGUID GUID);
```

Description

Call GetComputerForGUID to obtain the name of an available server machine. The GUID parameter (which specifies the global unique identifier of the desired server interface) is ignored.

GetComputerForGUID returns the ComputerName property of an available server from the Servers (see page 2481) property. If the LoadBalanced (see page 2480) property is true, this server is chosen at random from available servers. If the LoadBalanced (see page 2480) property is false, this server is the first available server listed in the Servers (see page 2481) property. If no servers are available, GetComputerForGUID raises an EBrokerException (see page 2466) exception.

A server is considered available if its Enabled property is true and its HasFailed property is false. The broker automatically sets HasFailed to false when it is notified of a connection failure for that server.

See Also

GetComputerForProgID (see page 2477)

ComputerName (see page 2472)

LoadBalanced (see page 2480)

EBrokerException (see page 2466)

Enabled (see page 2473)

HasFailed (see page 2474)

SetConnectStatus (see page 2479)

44.1.4.4.2 TSimpleObjectBroker.GetComputerForProgID Method

Returns a string that identifies an available server machine.

Pascal

```
function GetComputerForProgID(const ProgID): string; override;
```

C++

```
virtual __fastcall UnicodeString GetComputerForProgID(const  ProgID);
```

Description

Call GetComputerForProgID to obtain the name of an available server machine. The ProgID parameter (which specifies the desired server interface) is ignored.

TSimpleObjectBroker (see page 2475) returns the ComputerName property of an available server from the Servers (see page 2481) property. If the LoadBalanced (see page 2480) property is true, this server is chosen at random from available servers. If the LoadBalanced (see page 2480) property is false, this server is the first available server listed in the Servers (see page 2481) property. If no servers are available, GetComputerForGUID (see page 2477) raises an EBrokerException (see page 2466) exception.

A server is considered available if its Enabled property is true and its HasFailed property is false. The broker automatically sets HasFailed to false when it is notified of a connection failure for that server.

See Also

GetComputerForGUID (see page 2477)

ComputerName (see page 2472)

LoadBalanced (see page 2480)

EBrokerException (see page 2466)

Enabled (see page 2473)

HasFailed (see page 2474)

SetConnectStatus (see page 2479)

44.1.4.4.3 TSimpleObjectBroker.GetPortForComputer Method

Returns a port number for connecting to a specified remote server.

Pascal

```
function GetPortForComputer(const ComputerName: string): Integer; override;
```

C++

```
virtual __fastcall int GetPortForComputer(const UnicodeString ComputerName);
```

Description

Socket connection components call GetPortForComputer to get a value for their Port property. The ComputerName parameter specifies a Host name or IP address that was returned by GetComputerForGUID (see page 2477) or GetComputerForProgID (see page 2477). This value is used to identify a server in the list maintained by the Servers (see page 2481) property.

There are no connection components that explicitly use a Port number except on the Windows product. GetPortForComputer is included in TCustomObjectBroker (see page 1281) to allow for the possibility of connection components that use an explicit Port number. The ComputerName parameter is a Host name or IP address that is returned by the GetComputerForGUID (see page 2477) or GetComputerForProgID (see page 2477) method. This value is used to identify a server in the list maintained by the Servers (see page 2481) property.

See Also

GetComputerForProgID (see page 2477)

GetComputerForGUID (see page 2477)

Port

ComputerName (see page 2472)

Port (see page 2474)

44.1.4.4.4 TSimpleObjectBroker.LoadFromStream Method

Sets the value of the Servers (see page 2481) property from a stream.

Pascal

```
procedure LoadFromStream(Stream: TStream);
```

C++

```
__fastcall LoadFromStream(TStream Stream);
```

Description

Call LoadFromStream to load a list of available servers that has been previously saved using the SaveToStream (see page 2479) method.

See Also

SaveToStream (see page 2479)

ServerData (see page 2481)

44.1.4.4.5 TSimpleObjectBroker.SaveToStream Method

Saves the value of the Servers (see page 2481) property to a stream.

Pascal

```
procedure SaveToStream(Stream: TStream);
```

C++

```
__fastcall SaveToStream(TStream Stream);
```

Description

Call SaveToStream to save the list of available servers. The servers can be restored later by calling the LoadFromStream (see page 2479) method.

See Also

LoadFromStream (see page 2479)

ServerData (see page 2481)

44.1.4.4.6 TSimpleObjectBroker.SetConnectStatus Method

Records whether a connection was established to a specified server.

Pascal

```
procedure SetConnectStatus(ComputerName: string; Success: Boolean); override;
```

C++

```
virtual __fastcall SetConnectStatus(UnicodeString ComputerName, Boolean Success);
```

Description

After calling GetComputerForGUID (see page 2477) or GetComputerForProgID (see page 2477), connection components use the returned value to open a connection to the identified server. They then call SetConnectStatus to inform the broker of whether the connection was successfully opened.

SetConnectStatus sets the HasFailed property of the corresponding server to true if the connection was unsuccessful. It sets the HasFailed property to false if the connection was successful.

When the HasFailed property of a server is true, that server is ignored when the GetComputerForGUID (see page 2477) or GetComputerForProgID (see page 2477) method looks for available connections.

See Also

GetComputerForGUID (see page 2477)

GetComputerForProgID (see page 2477)

44.1.4.5 TSimpleObjectBroker Properties

The properties of the TSimpleObjectBroker class are listed here.

Public Properties

	Name	Description
	ServerData (see page 2481)	Encapsulates the list of servers as an OleVariant.

Published Properties

	Name	Description
	LoadBalanced (see page 2480)	Specifies how the simple object broker picks a server from its list.
	Servers (see page 2481)	Maintains the list of servers that can be provided to connection components.

44.1.4.5.1 TSimpleObjectBroker.LoadBalanced Property

Specifies how the simple object broker picks a server from its list.

Pascal

```
property LoadBalanced: Boolean;
```

C++

```
__property Boolean LoadBalanced;
```

Description

Set LoadBalanced to true to indicate that the simple object broker should supply computer names randomly from the available servers listed in the Servers (see page 2481) property. When LoadBalanced is false, GetComputerForGUID (see page 2477) and GetComputerForProgID (see page 2477) always return the first available server.

If the simple object broker is used by multiple connection components in the client application, it is preferable to set LoadBalanced to true in order to distribute the work among available servers.

When there is only a single connection component, there is no need to randomize the selection of servers, because only one connection will be used at a time. However, when deploying client applications, you may want to reorder the Servers (see page 2481) list in the simple object broker for each client, so that different clients give preference to different servers.

See Also

Servers (see page 2481)

GetComputerForGUID (see page 2477)

GetComputerForProgID (see page 2477)

44.1.4.5.2 TSimpleObjectBroker.ServerData Property

Encapsulates the list of servers as an OleVariant.

Pascal

```
property ServerData: OleVariant;
```

C++

```
__property OleVariant ServerData;
```

Description

Reading ServerData returns the value of the Servers (see page 2481) property stored in the internal component format. Writing ServerData sets the value of the Servers (see page 2481) property from an OleVariant that was saved previously.

Use the ServerData property to provide a mechanism for loading and saving the list of servers represented by the Servers (see page 2481) property. For example, An application can write the value of ServerData to an .ini file or the system registry. Later, it can read the value it saved as an OleVariant, and set ServerData to restore the Servers (see page 2481) property to the value that was saved.

Note: The implementation of ServerData uses the SaveToStream (see page 2479) method to encode the Servers (see page 2481) property as an OleVariant and the LoadFromStream (see page 2479) method to decode that OleVariant and restore the Servers (see page 2481) property.

See Also

Servers (see page 2481)

LoadFromStream (see page 2479)

SaveToStream (see page 2479)

44.1.4.5.3 TSimpleObjectBroker.Servers Property

Maintains the list of servers that can be provided to connection components.

Pascal

```
property Servers: TServerCollection;
```

C++

```
__property TServerCollection Servers;
```

Description

Use Servers to specify the list of servers that the simple object broker can return from the GetComputerForProgID (see page 2477) and GetComputerForGUID (see page 2477) methods.

Each item in the collection keeps track of whether the last connection attempt to that server was successful, and if not, the simple object broker ignores that server. In addition, individual servers in the collection can be disabled so that the simple object broker ignores them until they are re-enabled.

See Also

ServerData (see page 2481)

GetComputerForProgID (see page 2477)

GetComputerForGUID (see page 2477)

TServerItem (see page 2470)

45 ObjComAuto Namespace

This is namespace ObjComAuto.

Topics

Name	Description
Classes (see page 2484)	The following table lists classes in this documentation.
Functions (see page 2489)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 2491)	The following table lists structs, records, enums in this documentation.
Types (see page 2493)	The following table lists types in this documentation.

Classes

	Name	Description
	TObjectDispatch (see page 2484)	This is class ObjComAuto.TObjectDispatch.

Functions

	Name	Description
	CreateMethodPointer (see page 2489)	This is function ObjComAuto.CreateMethodPointer.
	CreateMethodPointer (see page 2489)	This is function ObjComAuto.CreateMethodPointer.
	ReleaseMethodPointer (see page 2490)	This is function ObjComAuto.ReleaseMethodPointer.

Structs, Records, Enums

	Name	Description
	TDispatchInfo (see page 2491)	This is record ObjComAuto.TDispatchInfo.
	TDispatchKind (see page 2491)	This is record ObjComAuto.TDispatchKind.

Types

Name	Description
TDispatchInfos (see page 2493)	This is type ObjComAuto.TDispatchInfos.

45.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TObjectDispatch (see page 2484)	This is class ObjComAuto.TObjectDispatch.

45.1.1 TObjectDispatch Class

Class Hierarchy

Pascal

```
TObjectDispatch = class(TInterfacedObject, IDispatch);
```

C++

```
class TObjectDispatch : public TInterfacedObject, public IDispatch;
```

File

ObjComAuto

Description

This is class ObjComAuto.TObjectDispatch.

45.1.1.1 TObjectDispatch Members

The following tables list the members exposed by TObjectDispatch.

Protected Methods

	Name	Description
V	GetMethodInfo (see page 2486)	This is GetMethodInfo, a member of class TObjectDispatch.
V	GetObjectDispatch (see page 2486)	This is GetObjectDispatch, a member of class TObjectDispatch.
V	GetPropInfo (see page 2487)	This is GetPropInfo, a member of class TObjectDispatch.

Protected Properties

	Name	Description
R	Instance (see page 2488)	This is Instance, a member of class TObjectDispatch.

Public Methods

	Name	Description
	Create (see page 2485)	This is Create, a member of class TObjectDispatch.
V	Destroy (see page 2485)	Disposes of an object instance.
V	GetIDsOfNames (see page 2486)	GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name.

	GetTypeInfo (see page 2487)	GetTypeInfo returns the object's type information.
	GetTypeInfoCount (see page 2487)	GetTypeInfoCount returns the count of type information interfaces.
	Invoke (see page 2488)	Invoke performs a call to a late-binded property or method.

45.1.1.2 TObjectDispatch.Create Constructor

Pascal

```
constructor Create(Instance: TObject; Owned: Boolean = True);
```

C++

```
__fastcall TObjectDispatch(TObject * Instance, Boolean Owned = True);
```

Description

This is Create, a member of class TObjectDispatch.

45.1.1.3 TObjectDispatch.Destroy Destructor

Disposes of an object instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TObjectDispatch();
```

Description

Do not call System::TObject::Destroy directly. Call System::TObject::Free instead. Free verifies that the object reference is not nil before calling System::TObject::Destroy.

The System::TObject::Destroy method defined by System::TObject deallocates memory. Descendant objects usually define a destructor that is customized for that particular kind of object.

When declaring a System::TObject::Destroy method in a descendant, always add the override directive to the declaration and call the inherited System::TObject::Destroy as the last statement in the overriding method. Since System::TObject::Destroy is a virtual method, overriding it ensures that the proper inherited behavior occurs.

Note: If an exception escapes from the constructor, the destructor is called to destroy the partially constructed object instance that failed to initialize completely. Therefore, destructors should check that allocated resources, such as handles, were actually allocated before trying to release them, since their value might be zero.

System::TObject::Destroy should be implemented so that it calls System::TObject::Free on all subobjects created within the object's constructor (that is, allocated by the constructor). Unlike System::TObject::Destroy, System::TObject::Free provides a safeguard when destroying objects that are nil.

See Also

System::TObject::Free

45.1.1.4 TObjectDispatch Methods

The methods of the TObjectDispatch class are listed here.

Protected Methods

	Name	Description
	GetMethodInfo (see page 2486)	This is GetMethodInfo, a member of class TObjectDispatch.
	GetObjectDispatch (see page 2486)	This is GetObjectDispatch, a member of class TObjectDispatch.
	GetPropInfo (see page 2487)	This is GetPropInfo, a member of class TObjectDispatch.

Public Methods

	Name	Description
	GetIDsOfNames (see page 2486)	GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name.
	GetTypeInfo (see page 2487)	GetTypeInfo returns the object's type information.
	GetTypeInfoCount (see page 2487)	GetTypeInfoCount returns the count of type information interfaces.
	Invoke (see page 2488)	Invoke performs a call to a late-binded property or method.

45.1.1.4.1 TObjectDispatch.GetIDsOfNames Method

GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name.

Pascal

```
function GetIDsOfNames(const IID: TGUID; Names: Pointer; NameCount: Integer; LocaleID:
Integer; DispIDs: Pointer): HRESULT; virtual; stdcall;
```

C++

```
virtual __fastcall __stdcall HRESULT GetIDsOfNames(const TGUID IID, void * Names, int
NameCount, int LocaleID, void * DispIDs);
```

Description

GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name. GetIDsOfNames is used primarily to obtain dispatch identifiers for the late-binded methods exposed by the COM object through the IDispatch interface.

See Also

GetTypeInfoCount

GetTypeInfo

Invoke

45.1.1.4.2 TObjectDispatch.GetMethodInfo Method

Pascal

```
function GetMethodInfo(const AName: ShortString; var AInstance: TObject): PMethodInfoHeader;
virtual;
```

C++

```
virtual __fastcall PMethodInfoHeader GetMethodInfo(const ShortString AName, TObject *
AInstance);
```

Description

This is GetMethodInfo, a member of class TObjectDispatch.

45.1.1.4.3 TObjectDispatch.GetObjectDispatch Method

Pascal

```
function GetObjectDispatch(Obj: TObject): TObjectDispatch; virtual;
```

C++

```
virtual __fastcall TObjectDispatch GetObjectDispatch(TObject * Obj);
```

Description

This is GetObjectDispatch, a member of class TObjectDispatch.

45.1.1.4.4 TObjectDispatch.GetPropInfo Method

Pascal

```
function GetPropInfo(const AName: string; var AInstance: TObject; var CompIndex: Integer):
PPropInfo; virtual;
```

C++

```
virtual __fastcall PPropInfo GetPropInfo(const UnicodeString AName, TObject * AInstance, int
CompIndex);
```

Description

This is GetPropInfo, a member of class TObjectDispatch.

45.1.1.4.5 TObjectDispatch.GetTypeInfo Method

GetTypeInfo returns the object's type information.

Pascal

```
function GetTypeInfo(Index: Integer; LocaleID: Integer; out TypeInfo): HRESULT; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetTypeInfo(int Index, int LocaleID,  TypeInfo);
```

Description

GetTypeInfo returns the type information for a specified index. The index is previously obtained by a call to the GetTypeInfoCount method. If the method returns S_OK, the `TypeInfo` parameter is set to point to the requested type information; otherwise an error code is returned.

See Also

GetTypeInfoCount

GetIDsOfNames

Invoke

45.1.1.4.6 TObjectDispatch.GetTypeInfoCount Method

GetTypeInfoCount returns the count of type information interfaces.

Pascal

```
function GetTypeInfoCount(out Count: Integer): HRESULT; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetTypeInfoCount(int Count);
```

Description

GetTypeInfoCount returns the count of type information interfaces supported by the implementing COM object. If the method returns S_OK, the `Count` parameter is set to the number of type information interfaces, which is either 1 (if the object supports

type information) or 0 (if the object does not support type information). A return value of E_NOTIMPL means that the method is not implemented in the COM object.

See Also

GetTypeInfo

GetIDsOfNames

Invoke

45.1.1.4.7 TObjectDispatch.Invoke Method

Invoke performs a call to a late-binded property or method.

Pascal

```
function Invoke(DispID: Integer; const IID: TGUID; LocaleID: Integer; Flags: Word; var Params;
VarResult: Pointer; ExcepInfo: Pointer; ArgErr: Pointer): HRESULT; virtual; stdcall;
```

C++

```
virtual __fastcall __stdcall HRESULT Invoke(int DispID, const TGUID IID, int LocaleID, Word
Flags,  Params, void * VarResult, void * ExcepInfo, void * ArgErr);
```

Description

Invoke actually calls a method or a property exposed by the COM object through IDispatch. Invoke requires a previously obtained dispatch identifier. The identifier uniquely identifies a method, a property, and a list of parameters to be passed to the called method and exception information. The dispatch identifier of a method or property (also known as **dispid**) can be obtained by a calling the GetIDsOfNames method.

See Also

GetTypeInfoCount

GetTypeInfo

GetIDsOfNames

45.1.1.5 TObjectDispatch Properties

The properties of the TObjectDispatch class are listed here.

Protected Properties

	Name	Description
	Instance (see page 2488)	This is Instance, a member of class TObjectDispatch.

45.1.1.5.1 TObjectDispatch.Instance Property

Pascal

```
property Instance: TObject;
```

C++

```
__property TObject * Instance;
```

Description

This is Instance, a member of class TObjectDispatch.

45.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	CreateMethodPointer (see page 2489)	This is function ObjComAuto.CreateMethodPointer.
	CreateMethodPointer (see page 2489)	This is function ObjComAuto.CreateMethodPointer.
	ReleaseMethodPointer (see page 2490)	This is function ObjComAuto.ReleaseMethodPointer.

45.2.1 ObjComAuto.CreateMethodPointer Function

Pascal

```
function CreateMethodPointer(const Instance: IDispatch; DispID: Integer; TypeInfo: PTypeInfo):
TMethod; overload;
```

C++

```
TMethod CreateMethodPointer(const IDispatch Instance, int DispID, PTypeInfo TypeInfo);
```

File

ObjComAuto

Description

This is function ObjComAuto.CreateMethodPointer.

45.2.2 ObjComAuto.CreateMethodPointer Function

Pascal

```
function CreateMethodPointer(const Instance: IDispatch; const MethodName: string; TypeInfo:
PTypeInfo): TMethod; overload;
```

C++

```
TMethod CreateMethodPointer(const IDispatch Instance, const UnicodeString MethodName,
PTypeInfo TypeInfo);
```

File

ObjComAuto

Description

This is function ObjComAuto.CreateMethodPointer.

45.2.3 ObjComAuto.ReleaseMethodPointer Function

Pascal

```
procedure ReleaseMethodPointer(MethodPointer: TMethod);
```

C++

```
ReleaseMethodPointer(TMethod MethodPointer);
```

File

ObjComAuto

Description

This is function ObjComAuto.ReleaseMethodPointer.

45.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TDispatchKind (see page 2491)	This is record ObjComAuto.TDispatchKind.

Records

	Name	Description
	TDispatchInfo (see page 2491)	This is record ObjComAuto.TDispatchInfo.

45.3.1 ObjComAuto.TDispatchInfo Record

Pascal

```
TDispatchInfo = record
  Instance: TObject;
  Kind: TDispatchKind;
  case TDispatchKind of
    dkMethod: (MethodInfo: PMethodInfoHeader;);
    dkProperty: (PropInfo: PPropInfo;);
    dkSubComponent: (Index: Integer;);
end;
```

C++

```
struct TDispatchInfo {
  TObject * Instance;
  TDispatchKind Kind;
  PMethodInfoHeader MethodInfo;
  PPropInfo PropInfo;
  int Index;
};
```

File

ObjComAuto

Description

This is record ObjComAuto.TDispatchInfo.

45.3.2 ObjComAuto.TDispatchKind Enumeration

Pascal

```
TDispatchKind = (
  dkMethod,
  dkProperty,
  dkSubComponent
);
```

C++

```
enum TDispatchKind {
  dkMethod,
  dkProperty,
  dkSubComponent
};
```

File

ObjComAuto

Description

This is record ObjComAuto.TDispatchKind.

45.4 Types

The following table lists types in this documentation.

Types

Name	Description
TDispatchInfos (see page 2493)	This is type ObjComAuto.TDispatchInfos.

45.4.1 ObjComAuto.TDispatchInfos Type

Pascal

```
TDispatchInfos = array of TDispatchInfo;
```

C++

```
array of TDispatchInfo TDispatchInfos;
```

File

ObjComAuto

Description

This is type ObjComAuto.TDispatchInfos.

46 OleAuto Namespace

This is namespace OleAuto.

Topics

Name	Description
Classes (see page 2497)	The following table lists classes in this documentation.
Functions (see page 2513)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 2521)	The following table lists structs, records, enums in this documentation.
Types (see page 2523)	The following table lists types in this documentation.
Variables (see page 2524)	The following table lists variables in this documentation.
Constants (see page 2525)	The following table lists constants in this documentation.

Classes

	Name	Description
	EOleError (see page 2497)	This is class OleAuto.EOleError.
	EOleException (see page 2497)	This is class OleAuto.EOleException.
	EOleSysError (see page 2499)	This is class OleAuto.EOleSysError.
	TAutoDispatch (see page 2500)	This is class OleAuto.TAutoDispatch.
	TAutomation (see page 2504)	This is class OleAuto.TAutomation.
	TAutoObject (see page 2507)	This is class OleAuto.TAutoObject.
	TRegistryClass (see page 2511)	This is class OleAuto.TRegistryClass.

Constants

Name	Description
MaxDispArgs (see page 2525)	This is constant OleAuto.MaxDispArgs.

Functions

	Name	Description
	ClassIDToProgID (see page 2513)	This is function OleAuto.ClassIDToProgID.
	ClassIDToString (see page 2513)	This is function OleAuto.ClassIDToString.
	CreateOleObject (see page 2514)	Instantiates an Automation (see page 2524) object.
	DispInvoke (see page 2515)	This is function OleAuto.DispInvoke.
	DispInvokeError (see page 2516)	This is function OleAuto.DispInvokeError.
	DllCanUnloadNow (see page 2516)	This is function OleAuto.DllCanUnloadNow.
	DllGetClassObject (see page 2516)	This is function OleAuto.DllGetClassObject.
	DllRegisterServer (see page 2517)	This is function OleAuto.DllRegisterServer.

	DllUnregisterServer (see page 2517)	This is function OleAuto.DllUnregisterServer.
	GetActiveOleObject (see page 2517)	This is function OleAuto.GetActiveOleObject.
	OleCheck (see page 2518)	This is function OleAuto.OleCheck.
	OleError (see page 2518)	This is function OleAuto.OleError.
	ProgIDToClassID (see page 2518)	This is function OleAuto.ProgIDToClassID.
	StringToClassID (see page 2519)	This is function OleAuto.StringToClassID.
	VarFromInterface (see page 2519)	This is function OleAuto.VarFromInterface.
	VarToAutoObject (see page 2519)	This is function OleAuto.VarToAutoObject.
	VarToInterface (see page 2520)	This is function OleAuto.VarToInterface.

Structs, Records, Enums

	Name	Description
	TAutoClassInfo (see page 2521)	This is record OleAuto.TAutoClassInfo.
	TAutoClassInstancing (see page 2521)	This is record OleAuto.TAutoClassInstancing.
	TStartMode (see page 2522)	This is record OleAuto.TStartMode.

Types

Name	Description
TAutoClass (see page 2523)	This is type OleAuto.TAutoClass.
TLastReleaseEvent (see page 2523)	This is type OleAuto.TLastReleaseEvent.

Variables

Name	Description
Automation (see page 2524)	Automation is for backward compatibility.

46.1 Classes

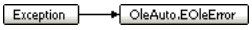
The following table lists classes in this documentation.

Classes

	Name	Description
	EOleError (see page 2497)	This is class OleAuto.EOleError.
	EOleException (see page 2497)	This is class OleAuto.EOleException.
	EOleSysError (see page 2499)	This is class OleAuto.EOleSysError.
	TAutoDispatch (see page 2500)	This is class OleAuto.TAutoDispatch.
	TAutomation (see page 2504)	This is class OleAuto.TAutomation.
	TAutoObject (see page 2507)	This is class OleAuto.TAutoObject.
	TRegistryClass (see page 2511)	This is class OleAuto.TRegistryClass.

46.1.1 EOleError Class

Class Hierarchy



Pascal

```
EOleError = class(Exception);
```

C++

```
class EOleError : public Exception;
```

File

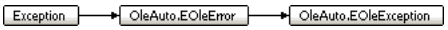
OleAuto

Description

This is class OleAuto.EOleError.

46.1.2 EOleException Class

Class Hierarchy



Pascal

```
EOleException = class(EOleError);
```

C++

```
class EOleException : public EOleError;
```

File

OleAuto

Description

This is class OleAuto.EOleException.

46.1.2.1 EOleException Members

The following tables list the members exposed by EOleException.

Public Methods

	Name	Description
	Create (see page 2498)	This is Create, a member of class EOleException.

Public Properties

	Name	Description
R	ErrorCode (see page 2498)	This is ErrorCode, a member of class EOleException.
R	HelpFile (see page 2499)	This is HelpFile, a member of class EOleException.
R	Source (see page 2499)	This is Source, a member of class EOleException.

46.1.2.2 EOleException.Create Constructor

Pascal

```
constructor Create(const ExcepInfo: TExcepInfo);
```

C++

```
__fastcall EOleException(const TExcepInfo ExcepInfo);
```

Description

This is Create, a member of class EOleException.

46.1.2.3 EOleException Properties

The properties of the EOleException class are listed here.

Public Properties

	Name	Description
R	ErrorCode (see page 2498)	This is ErrorCode, a member of class EOleException.
R	HelpFile (see page 2499)	This is HelpFile, a member of class EOleException.
R	Source (see page 2499)	This is Source, a member of class EOleException.

46.1.2.3.1 EOleException.ErrorCode Property

Pascal

```
property ErrorCode: Integer;
```

C++

```
__property int ErrorCode;
```

Description

This is ErrorCode, a member of class EOleException.

46.1.2.3.2 EOleException.HelpFile Property

Pascal

```
property HelpFile: string;
```

C++

```
__property UnicodeString HelpFile;
```

Description

This is HelpFile, a member of class EOleException.

46.1.2.3.3 EOleException.Source Property

Pascal

```
property Source: string;
```

C++

```
__property UnicodeString Source;
```

Description

This is Source, a member of class EOleException.

46.1.3 EOleSysError Class

Class Hierarchy

Pascal

```
EOleSysError = class(EOleError);
```

C++

```
class EOleSysError : public EOleError;
```

File

OleAuto

Description

This is class OleAuto.EOleSysError.

46.1.3.1 EOleSysError Members

The following tables list the members exposed by EOleSysError.

Public Methods

	Name	Description
	Create (see page 2500)	This is Create, a member of class EOleSysError.

Public Properties

	Name	Description
	ErrorCode (see page 2500)	This is ErrorCode, a member of class EOleSysError.

46.1.3.2 EOleSysError.Create Constructor

Pascal

```
constructor Create(ErrorCode: Integer);
```

C++

```
__fastcall EOleSysError(int ErrorCode);
```

Description

This is Create, a member of class EOleSysError.

46.1.3.3 EOleSysError Properties

The properties of the EOleSysError class are listed here.

Public Properties

	Name	Description
	ErrorCode (see page 2500)	This is ErrorCode, a member of class EOleSysError.

46.1.3.3.1 EOleSysError.ErrorCode Property

Pascal

```
property ErrorCode: Integer;
```

C++

```
__property int ErrorCode;
```

Description

This is ErrorCode, a member of class EOleSysError.

46.1.4 TAutoDispatch Class

Class Hierarchy

Pascal

```
TAutoDispatch = class(IDispatch);
```

C++

```
class TAutoDispatch : public IDispatch;
```

File

OleAuto

Description

This is class OleAuto.TAutoDispatch.

46.1.4.1 TAutoDispatch Members

The following tables list the members exposed by TAutoDispatch.

Public Methods

	Name	Description
	AddRef (see page 2502)	This is AddRef, a member of class TAutoDispatch.
	Create (see page 2501)	This is Create, a member of class TAutoDispatch.
	GetAutoObject (see page 2502)	This is GetAutoObject, a member of class TAutoDispatch.
	GetIDsOfNames (see page 2502)	This is GetIDsOfNames, a member of class TAutoDispatch.
	GetTypeInfo (see page 2502)	This is GetTypeInfo, a member of class TAutoDispatch.
	GetTypeInfoCount (see page 2502)	GetTypeInfoCount returns the count of type information interfaces.
	Invoke (see page 2503)	This is Invoke, a member of class TAutoDispatch.
	QueryInterface (see page 2503)	This is QueryInterface, a member of class TAutoDispatch.
	Release (see page 2503)	This is Release, a member of class TAutoDispatch.

Public Properties

	Name	Description
	AutoObject (see page 2504)	This is AutoObject, a member of class TAutoDispatch.

46.1.4.2 TAutoDispatch.Create Constructor

Pascal

```
constructor Create(AutoObject: TAutoObject);
```

C++

```
__fastcall TAutoDispatch(TAutoObject AutoObject);
```

Description

This is Create, a member of class TAutoDispatch.

46.1.4.3 TAutoDispatch Methods

The methods of the TAutoDispatch class are listed here.

Public Methods

	Name	Description
	AddRef (see page 2502)	This is AddRef, a member of class TAutoDispatch.
	GetAutoObject (see page 2502)	This is GetAutoObject, a member of class TAutoDispatch.
	GetIDsOfNames (see page 2502)	This is GetIDsOfNames, a member of class TAutoDispatch.
	GetTypeInfo (see page 2502)	This is GetTypeInfo, a member of class TAutoDispatch.
	GetTypeInfoCount (see page 2502)	GetTypeInfoCount returns the count of type information interfaces.
	Invoke (see page 2503)	This is Invoke, a member of class TAutoDispatch.
	QueryInterface (see page 2503)	This is QueryInterface, a member of class TAutoDispatch.
	Release (see page 2503)	This is Release, a member of class TAutoDispatch.

46.1.4.3.1 TAutoDispatch.AddRef Method

Pascal

```
function AddRef: Longint; override;
```

C++

```
virtual __fastcall Longint AddRef();
```

Description

This is AddRef, a member of class TAutoDispatch.

46.1.4.3.2 TAutoDispatch.GetAutoObject Method

Pascal

```
function GetAutoObject: TAutoObject; virtual; stdcall;
```

C++

```
virtual __fastcall __stdcall TAutoObject GetAutoObject();
```

Description

This is GetAutoObject, a member of class TAutoDispatch.

46.1.4.3.3 TAutoDispatch.GetIDsOfNames Method

Pascal

```
function GetIDsOfNames(const iid: TIID; rgszNames: POleStrList; cNames: Integer; lcid: TLCID;
rgdispid: PDispIDList): HResult; override;
```

C++

```
virtual __fastcall HRESULT GetIDsOfNames(const TIID iid, POleStrList rgszNames, int cNames,
TLCID lcid, PDispIDList rgdispid);
```

Description

This is GetIDsOfNames, a member of class TAutoDispatch.

46.1.4.3.4 TAutoDispatch.GetTypeInfo Method

Pascal

```
function GetTypeInfo(itinfo: Integer; lcid: TLCID; var tinfo: ITypeInfo): HResult; override;
```

C++

```
virtual __fastcall HRESULT GetTypeInfo(int itinfo, TLCID lcid, ITypeInfo tinfo);
```

Description

This is GetTypeInfo, a member of class TAutoDispatch.

46.1.4.3.5 TAutoDispatch.GetTypeInfoCount Method

GetTypeInfoCount returns the count of type information interfaces.

Pascal

```
function GetTypeInfoCount(var ctinfo: Integer): HResult; override;
```

C++

```
virtual __fastcall HRESULT GetTypeInfoCount(int ctinfo);
```

Description

GetTypeInfoCount returns the count of type information interfaces supported by the implementing COM object. If the method returns S_OK, the `Count` parameter is set to the number of type information interfaces, which is either 1 (if the object supports type information) or 0 (if the object does not support type information). A return value of E_NOTIMPL means that the method is not implemented in the COM object.

See Also

GetTypeInfo

GetIDsOfNames

Invoke

46.1.4.3.6 TAutoDispatch.Invoke Method

Pascal

```
function Invoke(dispIDMember: TDispID; const iid: TIID; lcid: TLCID; flags: Word; var
dispParams: TDispParams; varResult: PVariant; excepInfo: PExcepInfo; argErr: PInteger):
HResult; override;
```

C++

```
virtual __fastcall HRESULT Invoke(TDispID dispIDMember, const TIID iid, TLCID lcid, Word
flags, TDispParams dispParams, PVariant varResult, PExcepInfo excepInfo, PInteger argErr);
```

Description

This is Invoke, a member of class TAutoDispatch.

46.1.4.3.7 TAutoDispatch.QueryInterface Method

Pascal

```
function QueryInterface(const iid: TIID; var obj): HResult; override;
```

C++

```
virtual __fastcall HRESULT QueryInterface(const TIID iid,  obj);
```

Description

This is QueryInterface, a member of class TAutoDispatch.

46.1.4.3.8 TAutoDispatch.Release Method

Pascal

```
function Release: Longint; override;
```

C++

```
virtual __fastcall Longint Release();
```

Description

This is Release, a member of class TAutoDispatch.

46.1.4.4 TAutoDispatch Properties

The properties of the TAutoDispatch class are listed here.

Public Properties

	Name	Description
	AutoObject (see page 2504)	This is AutoObject, a member of class TAutoDispatch.

46.1.4.4.1 TAutoDispatch.AutoObject Property

Pascal

```
property AutoObject: TAutoObject;
```

C++

```
__property TAutoObject AutoObject;
```

Description

This is AutoObject, a member of class TAutoDispatch.

46.1.5 TAutomation Class

Class Hierarchy

OleAuto.TAutomation

Pascal

```
TAutomation = class;
```

C++

```
class TAutomation;
```

File

OleAuto

Description

This is class OleAuto.TAutomation.

46.1.5.1 TAutomation Members

The following tables list the members exposed by TAutomation.

Public Events

	Name	Description
	OnLastRelease (see page 2507)	This is OnLastRelease, a member of class TAutomation.

Public Methods

	Name	Description
	Create (see page 2505)	This is Create, a member of class TAutomation.
V	Destroy (see page 2505)	This is Destroy, a member of class TAutomation.
	RegisterClass (see page 2505)	This is RegisterClass, a member of class TAutomation.
	UpdateRegistry (see page 2506)	This is UpdateRegistry, a member of class TAutomation.

Public Properties

	Name	Description
R	AutoObjectCount (see page 2506)	This is AutoObjectCount, a member of class TAutomation.
	IsInprocServer (see page 2506)	This is IsInprocServer, a member of class TAutomation.
R	StartMode (see page 2507)	This is StartMode, a member of class TAutomation.

46.1.5.2 TAutomation.Create Constructor

Pascal

```
constructor Create;
```

C++

```
__fastcall TAutomation();
```

Description

This is Create, a member of class TAutomation.

46.1.5.3 TAutomation.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TAutomation();
```

Description

This is Destroy, a member of class TAutomation.

46.1.5.4 TAutomation Methods

The methods of the TAutomation class are listed here.

Public Methods

	Name	Description
	RegisterClass (see page 2505)	This is RegisterClass, a member of class TAutomation.
	UpdateRegistry (see page 2506)	This is UpdateRegistry, a member of class TAutomation.

46.1.5.4.1 TAutomation.RegisterClass Method

Pascal

```
procedure RegisterClass(const AutoClassInfo: TAutoClassInfo);
```

C++

```
__fastcall RegisterClass(const TAutoClassInfo AutoClassInfo);
```

Description

This is RegisterClass, a member of class TAutomation.

46.1.5.4.2 TAutomation.UpdateRegistry Method

Pascal

```
procedure UpdateRegistry(Register: Boolean);
```

C++

```
__fastcall UpdateRegistry(Boolean Register);
```

Description

This is UpdateRegistry, a member of class TAutomation.

46.1.5.5 TAutomation Properties

The properties of the TAutomation class are listed here.

Public Properties

	Name	Description
	AutoObjectCount (see page 2506)	This is AutoObjectCount, a member of class TAutomation.
	IsInprocServer (see page 2506)	This is IsInprocServer, a member of class TAutomation.
	StartMode (see page 2507)	This is StartMode, a member of class TAutomation.

46.1.5.5.1 TAutomation.AutoObjectCount Property

Pascal

```
property AutoObjectCount: Integer;
```

C++

```
__property int AutoObjectCount;
```

Description

This is AutoObjectCount, a member of class TAutomation.

46.1.5.5.2 TAutomation.IsInprocServer Property

Pascal

```
property IsInprocServer: Boolean;
```

C++

```
__property Boolean IsInprocServer;
```

Description

This is IsInprocServer, a member of class TAutomation.

46.1.5.5.3 TAutomation.StartMode Property

Pascal

```
property StartMode: TStartMode;
```

C++

```
__property TStartMode StartMode;
```

Description

This is StartMode, a member of class TAutomation.

46.1.5.6 TAutomation Events

The events of the TAutomation class are listed here.

Public Events

	Name	Description
	OnLastRelease (see page 2507)	This is OnLastRelease, a member of class TAutomation.

46.1.5.6.1 TAutomation.OnLastRelease Event

Pascal

```
property OnLastRelease: TLastReleaseEvent;
```

C++

```
__property TLastReleaseEvent OnLastRelease;
```

Description

This is OnLastRelease, a member of class TAutomation.

46.1.6 TAutoObject Class

Class Hierarchy

Pascal

```
TAutoObject = class(TObject);
```

C++

```
class TAutoObject : public TObject;
```

File

OleAuto

Description

This is class OleAuto.TAutoObject.

46.1.6.1 TAutoObject Members

The following tables list the members exposed by TAutoObject.

Protected Methods

	Name	Description
	CreateAutoDispatch (see page 2510)	This is CreateAutoDispatch, a member of class TAutoObject.
	GetExceptionInfo (see page 2510)	This is GetExceptionInfo, a member of class TAutoObject.

Public Methods

	Name	Description
	AddRef (see page 2509)	This is AddRef, a member of class TAutoObject.
	Create (see page 2508)	Constructs an object and initializes its data before the object is first used.
	Destroy (see page 2508)	Disposes of an object instance.
	Release (see page 2510)	This is Release, a member of class TAutoObject.

Public Properties

	Name	Description
	AutoDispatch (see page 2510)	This is AutoDispatch, a member of class TAutoObject.
	OleObject (see page 2511)	This is OleObject, a member of class TAutoObject.
	RefCount (see page 2511)	This is RefCount, a member of class TAutoObject.

46.1.6.2 TAutoObject.Create Constructor

Constructs an object and initializes its data before the object is first used.

Pascal

```
constructor Create; virtual;
```

C++

```
virtual __fastcall TAutoObject();
```

Description

System::TObject::Create constructs an object. The purpose, size, and behavior of objects differ greatly. The System::TObject::Create constructor defined by System::TObject allocates memory but does not initialize data.

Descendant objects usually define a constructor that creates the particular kind of object and initializes its data.

Note: If an exception escapes from a constructor, the object's destructor is called to clean up the failed instance.

See Also

System::TObject::Destroy

System::TObject::Free

46.1.6.3 TAutoObject.Destroy Destructor

Disposes of an object instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TAutoObject();
```

Description

Do not call System::TObject::Destroy directly. Call System::TObject::Free instead. Free verifies that the object reference is not nil before calling System::TObject::Destroy.

The System::TObject::Destroy method defined by System::TObject deallocates memory. Descendant objects usually define a destructor that is customized for that particular kind of object.

When declaring a System::TObject::Destroy method in a descendant, always add the override directive to the declaration and call the inherited System::TObject::Destroy as the last statement in the overriding method. Since System::TObject::Destroy is a virtual method, overriding it ensures that the proper inherited behavior occurs.

Note: If an exception escapes from the constructor, the destructor is called to destroy the partially constructed object instance that failed to initialize completely. Therefore, destructors should check that allocated resources, such as handles, were actually allocated before trying to release them, since their value might be zero.

System::TObject::Destroy should be implemented so that it calls System::TObject::Free on all subobjects created within the object's constructor (that is, allocated by the constructor). Unlike System::TObject::Destroy, System::TObject::Free provides a safeguard when destroying objects that are nil.

See Also

System::TObject::Free

46.1.6.4 TAutoObject Methods

The methods of the TAutoObject class are listed here.

Protected Methods

	Name	Description
	CreateAutoDispatch (see page 2510)	This is CreateAutoDispatch, a member of class TAutoObject.
	GetExceptionInfo (see page 2510)	This is GetExceptionInfo, a member of class TAutoObject.

Public Methods

	Name	Description
	AddRef (see page 2509)	This is AddRef, a member of class TAutoObject.
	Release (see page 2510)	This is Release, a member of class TAutoObject.

46.1.6.4.1 TAutoObject.AddRef Method

Pascal

```
function AddRef: Integer;
```

C++

```
__fastcall int AddRef();
```

Description

This is AddRef, a member of class TAutoObject.

46.1.6.4.2 TAutoObject.CreateAutoDispatch Method

Pascal

```
function CreateAutoDispatch: TAutoDispatch; virtual;
```

C++

```
virtual __fastcall TAutoDispatch CreateAutoDispatch();
```

Description

This is CreateAutoDispatch, a member of class TAutoObject.

46.1.6.4.3 TAutoObject.GetExceptionInfo Method

Pascal

```
procedure GetExceptionInfo(ExceptObject: TObject; var ExcepInfo: TExcepInfo); virtual;
```

C++

```
virtual __fastcall GetExceptionInfo(TObject * ExceptObject, TExcepInfo ExcepInfo);
```

Description

This is GetExceptionInfo, a member of class TAutoObject.

46.1.6.4.4 TAutoObject.Release Method

Pascal

```
function Release: Integer;
```

C++

```
__fastcall int Release();
```

Description

This is Release, a member of class TAutoObject.

46.1.6.5 TAutoObject Properties

The properties of the TAutoObject class are listed here.

Public Properties

	Name	Description
	AutoDispatch (see page 2510)	This is AutoDispatch, a member of class TAutoObject.
	OleObject (see page 2511)	This is OleObject, a member of class TAutoObject.
	RefCount (see page 2511)	This is RefCount, a member of class TAutoObject.

46.1.6.5.1 TAutoObject.AutoDispatch Property

Pascal

```
property AutoDispatch: TAutoDispatch;
```

C++

```
__property TAutoDispatch AutoDispatch;
```

Description

This is AutoDispatch, a member of class TAutoObject.

46.1.6.5.2 TAutoObject.OleObject Property

Pascal

```
property OleObject: Variant;
```

C++

```
__property Variant OleObject;
```

Description

This is OleObject, a member of class TAutoObject.

46.1.6.5.3 TAutoObject.RefCount Property

Pascal

```
property RefCount: Integer;
```

C++

```
__property int RefCount;
```

Description

This is RefCount, a member of class TAutoObject.

46.1.7 TRegistryClass Class

Class Hierarchy

Pascal

```
TRegistryClass = class;
```

C++

```
class TRegistryClass;
```

File

OleAuto

Description

This is class OleAuto.TRegistryClass.

46.1.7.1 TRegistryClass Members

The following tables list the members exposed by TRegistryClass.

Public Methods

	Name	Description
	Create (see page 2512)	This is Create, a member of class TRegistryClass.
	Destroy (see page 2512)	This is Destroy, a member of class TRegistryClass.
	UpdateRegistry (see page 2512)	This is UpdateRegistry, a member of class TRegistryClass.

46.1.7.2 TRegistryClass.Create Constructor

Pascal

```
constructor Create(const AutoClassInfo: TAutoClassInfo);
```

C++

```
__fastcall TRegistryClass(const TAutoClassInfo AutoClassInfo);
```

Description

This is Create, a member of class TRegistryClass.

46.1.7.3 TRegistryClass.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRegistryClass();
```

Description

This is Destroy, a member of class TRegistryClass.

46.1.7.4 TRegistryClass Methods

The methods of the TRegistryClass class are listed here.

Public Methods

	Name	Description
	UpdateRegistry (see page 2512)	This is UpdateRegistry, a member of class TRegistryClass.

46.1.7.4.1 TRegistryClass.UpdateRegistry Method

Pascal

```
procedure UpdateRegistry(Register: Boolean);
```

C++

```
__fastcall UpdateRegistry(Boolean Register);
```

Description

This is UpdateRegistry, a member of class TRegistryClass.

46.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	ClassIDToProgID (see page 2513)	This is function OleAuto.ClassIDToProgID.
	ClassIDToString (see page 2513)	This is function OleAuto.ClassIDToString.
	CreateOleObject (see page 2514)	Instantiates an Automation (see page 2524) object.
	DispInvoke (see page 2515)	This is function OleAuto.DispInvoke.
	DispInvokeError (see page 2516)	This is function OleAuto.DispInvokeError.
	DllCanUnloadNow (see page 2516)	This is function OleAuto.DllCanUnloadNow.
	DllGetClassObject (see page 2516)	This is function OleAuto.DllGetClassObject.
	DllRegisterServer (see page 2517)	This is function OleAuto.DllRegisterServer.
	DllUnregisterServer (see page 2517)	This is function OleAuto.DllUnregisterServer.
	GetActiveOleObject (see page 2517)	This is function OleAuto.GetActiveOleObject.
	OleCheck (see page 2518)	This is function OleAuto.OleCheck.
	OleError (see page 2518)	This is function OleAuto.OleError.
	ProgIDToClassID (see page 2518)	This is function OleAuto.ProgIDToClassID.
	StringToClassID (see page 2519)	This is function OleAuto.StringToClassID.
	VarFromInterface (see page 2519)	This is function OleAuto.VarFromInterface.
	VarToAutoObject (see page 2519)	This is function OleAuto.VarToAutoObject.
	VarToInterface (see page 2520)	This is function OleAuto.VarToInterface.

46.2.1 OleAuto.ClassIDToProgID Function

Pascal

```
function ClassIDToProgID(const ClassID: TCLSID): string;
```

C++

```
UnicodeString ClassIDToProgID(const TCLSID ClassID);
```

File

OleAuto

Description

This is function OleAuto.ClassIDToProgID.

46.2.2 OleAuto.ClassIDToString Function

Pascal

```
function ClassIDToString(const ClassID: TCLSID): string;
```

C++

```
UnicodeString ClassIDToString(const TCLSID ClassID);
```

File

OleAuto

Description

This is function OleAuto.ClassIDToString.

46.2.3 OleAuto.CreateOleObject Function

Instantiates an Automation (see page 2524) object.

Pascal

```
function CreateOleObject(const ClassName: string): Variant;
```

C++

```
Variant CreateOleObject(const UnicodeString ClassName);
```

File

OleAuto

Description

CreateOleObject creates a single uninitialized object of the class specified by the `ClassName` parameter. `ClassName` specifies the string representation of the Class ID (CLSID). CreateOleObject is used to create an object of a specified type when the CLSID is known and when the object is on a local or in-proc server. Only the objects that are not part of an aggregate are created using CreateOleObject.

Note: In Delphi code, CreateOleObject is called once to create each new single instance of a class. To create multiple instances of the same class, we recommend that you use a class factory.

Tip: As shown in the example, you can release the interface stored in the returned Variant by assigning the Unassigned constant to that Variant.

C++ Examples:

```
/*
The following code shows an example of how to create an Ole Object and how to
perform specific operations.
*/
void __fastcall TMyForm::ButtonClick(TObject *Sender)
{
    OleVariant WordApp, NewDoc;

    /* Creates a Microsoft Word application. */
    WordApp = CreateOleObject("Word.Application");

    /* Creates a new Microsoft Word document. */
    NewDoc = WordApp.OlePropertyGet("Documents").OleFunction("Add");

    /* Inserts the text 'Hello World!' in the document. */
    WordApp.OlePropertyGet("Selection").OleFunction("TypeText", "Hello World!");

    /* Saves the document on the disk. */
    NewDoc.OleFunction("SaveAs", "my_new_document.doc");
```

```
    /* Closes Microsoft Word. */
    WordApp.OleFunction("Quit");

    /* Releases the interface by assigning the Unassigned constant to the Variant variables. */
    NewDoc = Unassigned;
    WordApp = Unassigned;
}
```

Delphi Examples:

```
{
The following code shows an example of how to create an Ole Object and how to
perform specific operations.
}
procedure TForm1.ButtonClick(Sender: TObject);
var
  WordApp, NewDoc: Variant;
begin
  { Creates a Microsoft Word application. }
  WordApp := CreateOleObject('Word.Application');
  { Creates a new Microsoft Word document. }
  NewDoc := WordApp.Documents.Add;
  { Inserts the text 'Hello World!' in the document. }
  WordApp.Selection.TypeText('Hello World!');
  { Saves the document on the disk. }
  NewDoc.SaveAs('my_new_document.doc');
  { Closes Microsoft Word. }
  WordApp.Quit;
  { Releases the interface by assigning the Unassigned constant to the Variant variables. }
  NewDoc := Unassigned;
  WordApp := Unassigned;
end;
```

See Also

CreateComObject

CreateOleObject

Unassigned

Connecting to a Server

Controlling an Automation Server Using a Dispatch Interface

46.2.4 OleAuto.DispInvoke Function

Pascal

```
procedure DispInvoke(Dispatch: IDispatch; CallDesc: PCallDesc; DispIDs: PDispIDList; Params:
Pointer; Result: PVariant);
```

C++

```
DispInvoke(IDispatch Dispatch, PCallDesc CallDesc, PDispIDList DispIDs, void * Params,
PVariant Result);
```

File

OleAuto

Description

This is function OleAuto.DispInvoke.

46.2.5 OleAuto.DispInvokeError Function

Pascal

```
procedure DispInvokeError(Status: HResult; const ExcepInfo: TExcepInfo);
```

C++

```
DispInvokeError(HRESULT Status, const TExcepInfo ExcepInfo);
```

File

OleAuto

Description

This is function OleAuto.DispInvokeError.

46.2.6 OleAuto.DllCanUnloadNow Function

Pascal

```
function DllCanUnloadNow: HResult; stdcall;
```

C++

```
__stdcall HRESULT DllCanUnloadNow();
```

File

OleAuto

Description

This is function OleAuto.DllCanUnloadNow.

46.2.7 OleAuto.DllGetClassObject Function

Pascal

```
function DllGetClassObject(const CLSID: TCLSID; const IID: TIID; var Obj): HResult; stdcall;
```

C++

```
__stdcall HRESULT DllGetClassObject(const TCLSID CLSID, const TIID IID,  Obj);
```

File

OleAuto

Description

This is function OleAuto.DllGetClassObject.

46.2.8 OleAuto.DllRegisterServer Function

Pascal

```
function DllRegisterServer: HResult; stdcall;
```

C++

```
__stdcall HRESULT DllRegisterServer();
```

File

OleAuto

Description

This is function OleAuto.DllRegisterServer.

46.2.9 OleAuto.DllUnregisterServer Function

Pascal

```
function DllUnregisterServer: HResult; stdcall;
```

C++

```
__stdcall HRESULT DllUnregisterServer();
```

File

OleAuto

Description

This is function OleAuto.DllUnregisterServer.

46.2.10 OleAuto.GetActiveOleObject Function

Pascal

```
function GetActiveOleObject(const ClassName: string): Variant;
```

C++

```
Variant GetActiveOleObject(const UnicodeString ClassName);
```

File

OleAuto

Description

This is function OleAuto.GetActiveOleObject.

46.2.11 OleAuto.OleCheck Function

Pascal

```
procedure OleCheck(Result: HResult);
```

C++

```
OleCheck(HRESULT Result);
```

File

OleAuto

Description

This is function OleAuto.OleCheck.

46.2.12 OleAuto.OleError Function

Pascal

```
procedure OleError(ErrorCode: HResult);
```

C++

```
OleError(HRESULT ErrorCode);
```

File

OleAuto

Description

This is function OleAuto.OleError.

46.2.13 OleAuto.ProgIDToClassID Function

Pascal

```
function ProgIDToClassID(const ProgID: string): TCLSID;
```

C++

```
TCLSID ProgIDToClassID(const UnicodeString ProgID);
```

File

OleAuto

Description

This is function OleAuto.ProgIDToClassID.

46.2.14 OleAuto.StringToClassID Function

Pascal

```
function StringToClassID(const S: string): TCLSID;
```

C++

```
TCLSID StringToClassID(const UnicodeString S);
```

File

OleAuto

Description

This is function OleAuto.StringToClassID.

46.2.15 OleAuto.VarFromInterface Function

Pascal

```
function VarFromInterface(Unknown: IUnknown): Variant;
```

C++

```
Variant VarFromInterface(IUnknown Unknown);
```

File

OleAuto

Description

This is function OleAuto.VarFromInterface.

46.2.16 OleAuto.VarToAutoObject Function

Pascal

```
function VarToAutoObject(const V: Variant): TAutoObject;
```

C++

```
TAutoObject VarToAutoObject(const Variant V);
```

File

OleAuto

Description

This is function OleAuto.VarToAutoObject.

46.2.17 OleAuto.VarToInterface Function

Pascal

```
function VarToInterface(const V: Variant): IDispatch;
```

C++

```
IDispatch VarToInterface(const Variant V);
```

File

OleAuto

Description

This is function OleAuto.VarToInterface.

46.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TAutoClassInstancing (see page 2521)	This is record OleAuto.TAutoClassInstancing.
	TStartMode (see page 2522)	This is record OleAuto.TStartMode.

Records

	Name	Description
	TAutoClassInfo (see page 2521)	This is record OleAuto.TAutoClassInfo.

46.3.1 OleAuto.TAutoClassInfo Record

Pascal

```
TAutoClassInfo = record
  AutoClass: TAutoClass;
  ProgID: string;
  ClassID: string;
  Description: string;
  Instancing: TAutoClassInstancing;
end;
```

C++

```
struct TAutoClassInfo {
  TAutoClass AutoClass;
  UnicodeString ProgID;
  UnicodeString ClassID;
  UnicodeString Description;
  TAutoClassInstancing Instancing;
};
```

File

OleAuto

Description

This is record OleAuto.TAutoClassInfo.

46.3.2 OleAuto.TAutoClassInstancing Enumeration

Pascal

```
TAutoClassInstancing = (
  acInternal,
  acSingleInstance,
  acMultiInstance
);
```

C++

```
enum TAutoClassInstancing {
  acInternal,
  acSingleInstance,
  acMultiInstance
};
```

File

OleAuto

Description

This is record OleAuto.TAutoClassInstancing.

46.3.3 OleAuto.TStartMode Enumeration

Pascal

```
TStartMode = (
  smStandalone,
  smAutomation,
  smRegServer,
  smUnregServer
);
```

C++

```
enum TStartMode {
  smStandalone,
  smAutomation,
  smRegServer,
  smUnregServer
};
```

File

OleAuto

Description

This is record OleAuto.TStartMode.

46.4 Types

The following table lists types in this documentation.

Types

Name	Description
TAutoClass (see page 2523)	This is type OleAuto.TAutoClass.
TLastReleaseEvent (see page 2523)	This is type OleAuto.TLastReleaseEvent.

46.4.1 OleAuto.TAutoClass Type

Pascal

```
TAutoClass = class of TAutoObject;
```

C++

```
TAutoObject TAutoClass;
```

File

OleAuto

Description

This is type OleAuto.TAutoClass.

46.4.2 OleAuto.TLastReleaseEvent Type

Pascal

```
TLastReleaseEvent = procedure (var Shutdown: Boolean) of object;
```

C++

```
(var Shutdown: Boolean) ( TLastReleaseEvent)();
```

File

OleAuto

Description

This is type OleAuto.TLastReleaseEvent.

46.5 Variables

The following table lists variables in this documentation.

Variables

Name	Description
Automation (see page 2524)	Automation is for backward compatibility.

46.5.1 OleAuto.Automation Variable

Automation is for backward compatibility.

Pascal

```
Automation: TAutomation;
```

C++

```
TAutomation Automation;
```

File

OleAuto

Description

Declarations in the Oleauto unit are provided for backward compatibility with Delphi 2 and C++Builder 1. New implementations of OLE and ActiveX have replaced declarations in the Oleauto unit.

Automation is a global instance of the Automation manager class. It serves as the Automation manager for an OLE Automation server written with early versions of Delphi or C++Builder. The Automation manager handles all interactions between Automation objects that have been registered with it and the OLE Automation API.

46.6 Constants

The following table lists constants in this documentation.

Constants

Name	Description
MaxDispArgs (see page 2525)	This is constant OleAuto.MaxDispArgs.

46.6.1 OleAuto.MaxDispArgs Constant

Pascal

```
MaxDispArgs = 32;
```

C++

```
MaxDispArgs = 32;
```

File

OleAuto

Description

This is constant OleAuto.MaxDispArgs.

47 OleCtnrs Namespace

This is namespace OleCtnrs.

Topics

Name	Description
Classes (see page 2529)	The following table lists classes in this documentation.
Interfaces (see page 2615)	The following table lists interfaces in this documentation.
Functions (see page 2617)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 2618)	The following table lists structs, records, enums in this documentation.
Types (see page 2622)	The following table lists types in this documentation.
Constants (see page 2623)	The following table lists constants in this documentation.

Classes

	Name	Description
	TOleContainer (see page 2529)	TOleContainer embeds or links OLE objects.
	TOleForm (see page 2613)	TOleForm is an object that provides OLE support for a TForm object.

Constants

Name	Description
ovDiscardUndoState (see page 2623)	Specifies to discard the undo state.
ovHide (see page 2624)	Specifies to hide an OLE object.
ovInPlaceActivate (see page 2624)	Specifies the activation of an OLE object.
ovOpen (see page 2625)	Specifies the Open action of an OLE object.
ovPrimary (see page 2625)	Specifies the default action of an OLE object.
ovShow (see page 2626)	Specifies to show an OLE object.
ovUIActivate (see page 2627)	Specifies the activation of the UI(User Interface) of an OLE object.

Functions

	Name	Description
	DestroyMetaPict (see page 2617)	Frees a metafile resource given its handle.

Interfaces

	Name	Description
	IVCLFrameForm (see page 2615)	This is class OleCtnrs.IVCLFrameForm.

Structs, Records, Enums

	Name	Description
	TAutoActivate (see page 2618)	This is record OleCtnrs.TAutoActivate.
	TCreateInfo (see page 2618)	TCreateInfo is used by TOleContainer (see page 2529)'s CreateObjectFromInfo method.
	TCreateType (see page 2619)	TCreateType is an enumerated type that defines values for TCreateInfo (see page 2618)'s CreateType field.
	TObjectState (see page 2620)	This is record OleCtnrs.TObjectState.
	TSizeMode (see page 2621)	This is record OleCtnrs.TSizeMode.
	TVerbInfo (see page 2621)	This is record OleCtnrs.TVerbInfo.

Types

Name	Description
TObjectMoveEvent (see page 2622)	This is type OleCtnrs.TObjectMoveEvent.

47.1 Classes

The following table lists classes in this documentation.

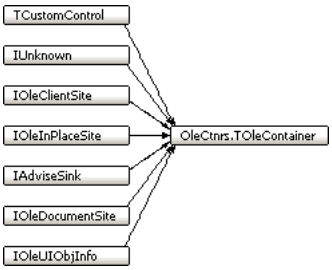
Classes

	Name	Description
	TOleContainer (see page 2529)	TOleContainer embeds or links OLE objects.
	TOleForm (see page 2613)	TOleForm is an object that provides OLE support for a TForm object.

47.1.1 TOleContainer Class

TOleContainer embeds or links OLE objects.

Class Hierarchy



Pascal

```
TOleContainer = class(TCustomControl, IUnknown, IOleClientSite, IOleInPlaceSite, IAdviseSink,
IOleDocumentSite, IOleUIObjInfo);
```

C++

```
class TOleContainer : public TCustomControl, public IUnknown, public IOleClientSite, public
IOleInPlaceSite, public IAdviseSink, public IOleDocumentSite, public IOleUIObjInfo;
```

File

OleCtnrs

Description

Use TOleContainer to handle many of the complexities of OLE 2.0. TOleContainer lets the user choose an OLE object to insert by simply calling the InsertObjectDialog (see page 2543) method. TOleContainer can create either an embedded OLE object or a linked OLE object.

TOleContainer automatically handles menu merging -- the process of combining the container form's menu with that of an in-place activated OLE object's server application. The menu items' GroupIndex property settings control how menus are merged. Those main menu items with GroupIndex values of 0, 2, and 4 remain; TOleContainer merges the server application's menus and replaces the main menu items with GroupIndex values of 1, 3, and 5 (if they exist).

OLE objects that are activated in-place add their server's toolbars directly into the container application's window. Normally, any panels used for toolbars are replaced by the OLE object's server's toolbars. Prevent this by setting a panel's Locked property to true.

When using TOleContainer in an SDI application (the main form's FormStyle property is fsNormal rather than fsMDIForm), place the TOleContainer component inside a panel whose Align (see page 2552) property is set to alClient. TOleContainer replaces an SDI form's toolbars as described above, and using a panel automatically adjusts the amount of space available to the OLE container.

See Also

TForm

TMainMenu (see page 1322)

TMenuItem (see page 1363)

TOleControl (see page 2638)

TPanel

Creating Clients for Servers That Do Not Have a Type Library

Active Documents

Building COM Applications

47.1.1.1 TOleContainer Members

The following tables list the members exposed by TOleContainer.

Protected Methods

	Name	Description
	_AddRef (see page 2534)	Called when an application uses a component interface.
	_Release (see page 2535)	Called when an application releases a component interface.
	Changed (see page 2535)	Sends a CM_CHANGED message to the control.
V	CreateParams (see page 2540)	Initializes a window-creation parameter data structure.
V	DblClick (see page 2540)	OnDblClick event dispatcher.
V	DefineProperties (see page 2541)	Provides methods to read and write the IsControl property to a stream such as a form file.
V	DoEnter (see page 2542)	Respond to receiving input focus.
V	GetPopupMenu (see page 2543)	Returns the pop-up menu associated with the control.
V	KeyDown (see page 2543)	Respond to key press events.
V	MouseDown (see page 2545)	OnMouseDown event dispatcher.
V	Paint (see page 2546)	Renders the image of a custom control.
V	QueryInterface (see page 2547)	Returns a reference to a specified interface if the object supports that interface.

Public Methods

	Name	Description
	ChangeIconDialog (see page 2535)	Displays the Change Icon OLE dialog box.
	Close (see page 2536)	Deactivates the OLE object.
	Copy (see page 2536)	Copies the OLE object to the Windows clipboard.
V	Create (see page 2532)	Creates and initializes an instance of TOleContainer (see page 2529).
	CreateLinkToFile (see page 2537)	Creates a linked OLE object from the contents of a file.
	CreateObject (see page 2537)	Creates an embedded OLE object given its class name (also known as programmatic identifier).
	CreateObjectFromFile (see page 2538)	Creates an embedded OLE object from the contents of a file.
	CreateObjectFromInfo (see page 2539)	Creates an OLE object matching the specified TCreateInfo (see page 2618) record.
V	Destroy (see page 2533)	Destroys an instance of TOleContainer (see page 2529).
	DestroyObject (see page 2541)	Destroys the container's OLE object.
	DoVerb (see page 2542)	Requests that the OLE object perform some action.

	GetIconMetaPict (see page 2542)	Returns the global memory handle of a metafile containing the OLE object's icon.
	InsertObjectDialog (see page 2543)	Displays the Insert Object OLE dialog box.
	LoadFromFile (see page 2544)	Loads an OLE object from the specified file.
	LoadFromStream (see page 2545)	Loads an OLE object from the specified stream.
	ObjectPropertiesDialog (see page 2546)	Displays the OLE Object Properties property sheet.
	Paste (see page 2546)	Pastes the contents of the Windows clipboard as an embedded object.
	PasteSpecialDialog (see page 2547)	Displays the Paste (see page 2546) Special OLE dialog box.
	Run (see page 2548)	Runs the OLE object's server application, but doesn't open or activate the OLE object itself.
	SaveAsDocument (see page 2548)	Saves the OLE object to the specified file in native OLE Document format.
	SaveToFile (see page 2550)	Saves the OLE object to the specified file.
	SaveToStream (see page 2550)	Saves the OLE object to the specified stream.
	UpdateObject (see page 2551)	Refreshes the OLE object.
	UpdateVerbs (see page 2551)	Refreshes the list of verbs the OLE object can respond to.

Public Properties

	Name	Description
R	CanPaste (see page 2558)	Indicates whether the Windows clipboard contains an OLE object that can be pasted into the OLE container.
R	Linked (see page 2567)	Indicates whether the OLE object is linked or embedded.
	Modified (see page 2567)	Indicates whether the OLE object has been modified.
R	NewInserted (see page 2567)	Indicates whether the OLE object was created rather than pasted in or created as a link to a file.
R	ObjectVerbs (see page 2568)	Lists the verbs that the OLE object supports.
R	OleClassName (see page 2569)	Specifies the class name of the OLE object.
R	OleObject (see page 2569)	Represents the OLE object hosted by the OLE container.
R	OleObjectInterface (see page 2569)	Provides access to the IOleObject interface for the OLE object.
R	PrimaryVerb (see page 2576)	Specifies the index of the primary (default) verb for the OLE object.
R	SourceDoc (see page 2579)	Specifies the name of the source document for a linked OLE object.
R	State (see page 2579)	Describes the state of the OLE object.
R	StorageInterface (see page 2579)	Provides access to the IStorage interface for the OLE object.

Published Events

	Name	Description
	OnActivate (see page 2583)	Occurs immediately after the OLE object is activated.
	OnDeactivate (see page 2583)	Occurs immediately after the OLE object is deactivated.
	OnDragDrop (see page 2584)	Occurs when the user drops an object being dragged.
	OnDragOver (see page 2587)	Occurs when the user drags an object over a control.
	OnEndDrag (see page 2590)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.
	OnEnter (see page 2591)	Occurs when a control receives the input focus.
	OnExit (see page 2593)	Occurs when the input focus shifts away from one control to another.
	OnKeyDown (see page 2597)	Occurs when a user presses any key while the control has focus.
	OnKeyPress (see page 2601)	Occurs when a key is pressed.
	OnKeyUp (see page 2602)	Occurs when the user releases a key that has been pressed.
	OnMouseActivate (see page 2604)	Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.
	OnMouseDown (see page 2605)	Occurs when the user presses a mouse button with the mouse pointer over a control.
	OnMouseMove (see page 2607)	Occurs when the user moves the mouse pointer while the mouse pointer is over a control.
	OnMouseUp (see page 2609)	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.
	OnObjectMove (see page 2611)	Occurs when the OLE container is notified that the OLE object has moved or resized.
	OnResize (see page 2612)	Occurs immediately after the OLE container window is resized.

	OnStartDrag (see page 2612)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

Published Properties

	Name	Description
	Align (see page 2552)	Determines how the control aligns within its container (parent control).
	AllowActiveDoc (see page 2556)	Specifies whether the OLE container supports the IOleDocumentSite interface.
	AllowInPlace (see page 2556)	Determines whether the OLE container allows in-place activation.
	Anchors (see page 2557)	Specifies how the control is anchored to its parent.
	AutoActivate (see page 2557)	Determines how the object in an OLE container is activated.
	AutoVerbMenu (see page 2558)	Determines whether the OLE container automatically creates a pop-up menu containing the OLE object's verbs.
	BorderStyle (see page 2558)	Determines whether a single line is drawn around the OLE container.
	Caption (see page 2559)	Specifies a text string that identifies the control to the user.
	Color (see page 2561)	Specifies the background color of the control.
	Constraints (see page 2562)	Specifies the size constraints for the control.
	CopyOnSave (see page 2563)	Determines whether the SaveToFile (see page 2550) and SaveToStream (see page 2550) methods first create a copy of the OLE object.
	Ctl3D (see page 2563)	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
	DragCursor (see page 2564)	Indicates the image used to represent the mouse pointer when the control is being dragged.
	DragMode (see page 2564)	Determines how the control initiates drag-and-drop or drag-and-dock operations.
	Enabled (see page 2566)	Controls whether the control responds to mouse, keyboard, and timer events.
	Iconic (see page 2566)	Determines whether the OLE object is displayed as an icon.
	OldStreamFormat (see page 2568)	Indicates whether the OLE object is loaded and saved using the VCL format.
	ParentColor (see page 2570)	Determines where a control looks for its color information.
	ParentCtl3D (see page 2570)	Determines where a component looks to determine if it should appear three dimensional.
	ParentShowHint (see page 2571)	Determines where a control looks to find out if its Help Hint should be shown.
	PopupMenu (see page 2571)	Identifies the pop-up menu associated with the control.
	ShowHint (see page 2577)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	SizeMode (see page 2578)	Determines how the OLE object is sized in the OLE container.
	TabOrder (see page 2580)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 2580)	Determines if the user can tab to a control.
	Visible (see page 2581)	Determines whether the component appears onscreen.

47.1.1.2 TOleContainer.Create Constructor

Creates and initializes an instance of TOleContainer (see page 2529).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TOleContainer(TComponent * AOwner);
```

Description

Call Create to instantiate a TOleContainer (see page 2529) object at runtime. OLE containers placed on forms at design time are created automatically.

Pass a component to the constructor to supply the OLE container with an Owner that is responsible for freeing it.

After calling the inherited constructor, Create

Sets ControlStyle to [csClickEvents, csSetCaption, csOpaque, csDoubleClicks].

Sets TabStop (see page 2580) to true.

Sets ParentColor (see page 2570) to false and BorderStyle (see page 2558) to bsSingle.

Sets AllowActiveDoc (see page 2556) and AllowInPlace (see page 2556) to true.

Sets AutoActivate (see page 2557) to aaDoubleClick and AutoVerbMenu (see page 2558) to true.

Sets CopyOnSave (see page 2563) to true.

See Also

AllowActiveDoc (see page 2556)

AllowInPlace (see page 2556)

AutoActivate (see page 2557)

AutoVerbMenu (see page 2558)

BorderStyle (see page 2558)

ControlStyle

CopyOnSave (see page 2563)

ParentColor

TabStop

47.1.1.3 TOleContainer.Destroy Destructor

Destroys an instance of TOleContainer (see page 2529).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TOleContainer();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TOleContainer (see page 2529) reference is not nil, and only then calls Destroy.Destroy frees the OLE object before calling the inherited destructor.

See Also

DestroyObject (see page 2541)

System::TObject::Free

47.1.1.4 TOleContainer Methods

The methods of the TOleContainer class are listed here.

Protected Methods

	Name	Description
	_AddRef (see page 2534)	Called when an application uses a component interface.
	_Release (see page 2535)	Called when an application releases a component interface.
	Changed (see page 2535)	Sends a CM_CHANGED message to the control.
V	CreateParams (see page 2540)	Initializes a window-creation parameter data structure.

	DblClick (see page 2540)	OnDblClick event dispatcher.
	DefineProperties (see page 2541)	Provides methods to read and write the IsControl property to a stream such as a form file.
	DoEnter (see page 2542)	Respond to receiving input focus.
	GetPopupMenu (see page 2543)	Returns the pop-up menu associated with the control.
	KeyDown (see page 2543)	Respond to key press events.
	MouseDown (see page 2545)	OnMouseDown event dispatcher.
	Paint (see page 2546)	Renders the image of a custom control.
	QueryInterface (see page 2547)	Returns a reference to a specified interface if the object supports that interface.

Public Methods

	Name	Description
	ChangeIconDialog (see page 2535)	Displays the Change Icon OLE dialog box.
	Close (see page 2536)	Deactivates the OLE object.
	Copy (see page 2536)	Copies the OLE object to the Windows clipboard.
	CreateLinkToFile (see page 2537)	Creates a linked OLE object from the contents of a file.
	CreateObject (see page 2537)	Creates an embedded OLE object given its class name (also known as programmatic identifier).
	CreateObjectFromFile (see page 2538)	Creates an embedded OLE object from the contents of a file.
	CreateObjectFromInfo (see page 2539)	Creates an OLE object matching the specified TCreateInfo (see page 2618) record.
	DestroyObject (see page 2541)	Destroys the container's OLE object.
	DoVerb (see page 2542)	Requests that the OLE object perform some action.
	GetIconMetaPict (see page 2542)	Returns the global memory handle of a metafile containing the OLE object's icon.
	InsertObjectDialog (see page 2543)	Displays the Insert Object OLE dialog box.
	LoadFromFile (see page 2544)	Loads an OLE object from the specified file.
	LoadFromStream (see page 2545)	Loads an OLE object from the specified stream.
	ObjectPropertiesDialog (see page 2546)	Displays the OLE Object Properties property sheet.
	Paste (see page 2546)	Pastes the contents of the Windows clipboard as an embedded object.
	PasteSpecialDialog (see page 2547)	Displays the Paste (see page 2546) Special OLE dialog box.
	Run (see page 2548)	Runs the OLE object's server application, but doesn't open or activate the OLE object itself.
	SaveAsDocument (see page 2548)	Saves the OLE object to the specified file in native OLE Document format.
	SaveToFile (see page 2550)	Saves the OLE object to the specified file.
	SaveToStream (see page 2550)	Saves the OLE object to the specified stream.
	UpdateObject (see page 2551)	Refreshes the OLE object.
	UpdateVerbs (see page 2551)	Refreshes the list of verbs the OLE object can respond to.

47.1.1.4.1 TOleContainer._AddRef Method

Called when an application uses a component interface.

Pascal

```
function _AddRef: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _AddRef();
```

Description

__AddRef is a basic implementation of the IInterface method, _AddRef.

Note: If the component is a wrapper for a COM object, __AddRef calls the _AddRef method of that COM object, and returns the resulting reference count.

In all other cases, __AddRef simply returns –1 and takes no action. This allows the component to implement interfaces where reference counting is not required. More sophisticated components should override __AddRef to implement reference counting.

See Also

QueryInterface

_Release

_AddRef

47.1.1.4.2 TOleContainer._Release Method

Called when an application releases a component interface.

Pascal

```
function _Release: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _Release();
```

Description

__Release is a basic implementation of the IInterface method, _Release.

__Release returns the resulting value of the reference count for the component's interface.

Note:

In all other cases, __Release simply returns –1 and takes no action. This allows the component to implement interfaces where reference counting is not required. More sophisticated components should override __Release to implement reference counting.

See Also

QueryInterface

_AddRef

47.1.1.4.3 TOleContainer.Changed Method

Sends a CM_CHANGED message to the control.

Pascal

```
procedure Changed; dynamic;
```

C++

```
__fastcall Changed();
```

Description

Call Changed after changing the properties of the control in a way that affects the properties of a parent object. Changed sends a CM_CHANGED message back to the control. If the control is a windowed control, this message is bubbled up to the parent so that it can make any necessary adjustments.

See Also

Parent

47.1.1.4.4 TOleContainer.ChangeIconDialog Method

Displays the Change Icon OLE dialog box.

Pascal

```
function ChangeIconDialog: Boolean;
```

C++

```
__fastcall Boolean ChangeIconDialog();
```

Description

Call ChangeIconDialog to let the user choose a different icon (from .ICO, .DLL, and .EXE files) to use when displaying the OLE object as an icon. An OLE object must already be loaded before calling ChangeIconDialog. ChangeIconDialog returns true if the dialog box was successfully displayed and the user chose the OK button. ChangeIconDialog returns false otherwise.

See Also

Iconic (see page 2566)

ObjectPropertiesDialog (see page 2546)

47.1.1.4.5 TOleContainer.Close Method

Deactivates the OLE object.

Pascal

```
procedure Close;
```

C++

```
__fastcall Close();
```

Description

Call Close to deactivate the OLE object and terminate its server application, without removing it from the container. Any changes the user made to the OLE object are automatically saved. After calling Close, the container's State (see page 2579) property is osLoaded.

Note: If Close is called when no OLE object is loaded, TOleContainer (see page 2529) raises an exception.

See Also

OnDeactivate (see page 2583)

State (see page 2579)

47.1.1.4.6 TOleContainer.Copy Method

Copies the OLE object to the Windows clipboard.

Pascal

```
procedure Copy;
```

C++

```
__fastcall Copy();
```

Description

Call Copy to write the OLE object to the clipboard. If there is no OLE object in the container, Copy raises an exception.

See Also

Paste (see page 2546)

PasteSpecialDialog (see page 2547)

SaveToFile (see page 2550)

SaveToStream (see page 2550)

47.1.1.4.7 TOleContainer.CreateLinkToFile Method

Creates a linked OLE object from the contents of a file.

Pascal

```
procedure CreateLinkToFile(const FileName: string; Iconic: Boolean);
```

C++

```
__fastcall CreateLinkToFile(const UnicodeString FileName, Boolean Iconic);
```

Description

Call CreateLinkToFile to create a linked OLE object from the file specified by the FileName parameter. Iconic (see page 2566) specifies whether the object is shown as an icon (true) or displayed as it would be in the server application (false). If there is already an OLE object in the container, it is destroyed and any changes the user made to it are discarded.

See Also

CreateObject (see page 2537)

CreateObjectFromFile (see page 2538)

CreateObjectFromInfo (see page 2539)

Creating Clients for Servers That Do Not Have a Type Library

47.1.1.4.8 TOleContainer.CreateObject Method

Creates an embedded OLE object given its class name (also known as programmatic identifier).

Pascal

```
procedure CreateObject(const OleClassName: string; Iconic: Boolean);
```

C++

```
__fastcall CreateObject(const UnicodeString OleClassName, Boolean Iconic);
```

Description

Call CreateObject to create an OLE object using the class name (also known as programmatic identifier) specified by the OleClassName (see page 2569) parameter. The Iconic (see page 2566) parameter indicates whether the object is shown as an icon (true) or displayed as it would be in the server application (false). If there's already an OLE object in the container, it is destroyed and any changes the user made to it are discarded.

See Also

CreateLinkToFile (see page 2537)

CreateObjectFromFile (see page 2538)

CreateObjectFromInfo (see page 2539)

OleClassName (see page 2569)

Creating Clients for Servers That Do Not Have a Type Library

47.1.1.4.9 TOleContainer.CreateObjectFromFile Method

Creates an embedded OLE object from the contents of a file.

Pascal

```
procedure CreateObjectFromFile(const FileName: string; Iconic: Boolean);
```

C++

```
__fastcall CreateObjectFromFile(const UnicodeString FileName, Boolean Iconic);
```

Description

Call CreateObjectFromFile to create an OLE object from the file specified by the FileName parameter. The Iconic (see page 2566) parameter indicates whether the object is shown as an icon (true) or displayed as it would be in the server application (false). If there's already an OLE object in the container, it is destroyed and any changes the user made to it are discarded.

C++ Examples:

```
/*
This example uses an OLE container and a main menu on a form.
When the user clicks the Open or Save options in the main menu,
an open/save dialog is displayed and by default,
the user is required to enter the name of a Word document.
Files with other extensions supported by the OLE container
can also be accessed. An additional test is made to assure that
the file to open exists and that no file overwriting is
going to take place.
*/
void __fastcall TForm1::Exit1Click(TObject *Sender)
{
  Close();
}

void __fastcall TForm1::Open1Click(TObject *Sender)
{
  if (OpenDialog1->Execute())
    if (FileExists(OpenDialog1->FileName))
      OleContainer1->CreateObjectFromFile(OpenDialog1->FileName, false);
    else
      throw(Exception("File not found."));
}

void __fastcall TForm1::Save1Click(TObject *Sender)
{
  if (SaveDialog1->Execute())
    if (FileExists(SaveDialog1->FileName))
      throw(Exception("File already exists. Cannot overwrite."));
  else
    OleContainer1->SaveAsDocument(SaveDialog1->FileName);
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  OpenDialog1->Filter = "Documents (*.doc)|*.DOC|Any file (*.*)|*.*";
  SaveDialog1->Filter = "Documents (*.doc)|*.DOC|Any file (*.*)|*.*";
}
```

Delphi Examples:

```
{
This example uses an OLE container and a main menu on a form.
When the user clicks the Open or Save options in the main menu,
an open/save dialog is displayed and by default,
the user is required to enter the name of a Word document.
```

```
  Files with other extensions supported by the OLE container
  can also be accessed. An additional test is made to assure that
  the file to open exists and that no file overwriting is
  going to take place.
  }
  procedure TForm1.Exit1Click(Sender: TObject);
  begin
    Close;
  end;

  procedure TForm1.FormCreate(Sender: TObject);
  begin
    OpenDialog1.Filter := 'Documents (*.doc)|*.DOC|Any file (*.*)|*.*';
    SaveDialog1.Filter := 'Documents (*.doc)|*.DOC|Any file (*.*)|*.*';
  end;

  procedure TForm1.Open1Click(Sender: TObject);
  begin
    if OpenDialog1.Execute then
      if FileExists(OpenDialog1.FileName) then
        OleContainer1.CreateObjectFromFile(OpenDialog1.FileName, false)
      else
        raise Exception.Create('File not found.');
  end;

  procedure TForm1.Save1Click(Sender: TObject);
  begin
    if SaveDialog1.Execute then
      if FileExists(SaveDialog1.FileName) then
        raise Exception.Create('File already exists. Cannot overwrite.')
      else
        OleContainer1.SaveAsDocument(SaveDialog1.FileName);
  end;
```

See Also

CreateLinkToFile (see page 2537)

CreateObject (see page 2537)

CreateObjectFromInfo (see page 2539)

Creating Clients for Servers That Do Not Have a Type Library

47.1.1.4.10 TOleContainer.CreateObjectFromInfo Method

Creates an OLE object matching the specified TCreateInfo (see page 2618) record.

Pascal

```
procedure CreateObjectFromInfo(const CreateInfo: TCreateInfo);
```

C++

```
__fastcall CreateObjectFromInfo(const TCreateInfo CreateInfo);
```

Description

Call CreateObjectFromInfo to create a linked or embedded OLE object based on the specifications in the CreateInfo parameter. CreateInfo specifies whether the object should be created from a file, from a data transfer object, or from a class identifier, and provides the necessary supporting information. CreateInfo also indicates whether the OLE object should be displayed as an Icon, and allows the icon to be specified. If there's already an OLE object in the container, it is destroyed and any changes the user made to it are discarded.

Note: The CreateLinkToFile (see page 2537), CreateObject (see page 2537), and CreateObjectFromFile (see page 2538) methods are one-line calls for the most common types of OLE objects. They can be used instead of filling out the

TCreateInfo (see page 2618) record.

See Also

CreateLinkToFile (see page 2537)

CreateObject (see page 2537)

CreateObjectFromFile (see page 2538)

Creating Clients for Servers That Do Not Have a Type Library

47.1.1.4.11 TOleContainer.CreateParams Method

Initializes a window-creation parameter data structure.

Pascal

```
procedure CreateParams(var Params: TCreateParams); override;
```

C++

```
virtual __fastcall CreateParams(TCreateParams Params);
```

Description

The CreateWnd method calls CreateParams to initialize the parameters it passes to CreateWindowHandle. Override CreateParams to customize the way a control creates its Windows representation. When overriding CreateParams, always call the inherited method first to set the default values, then make any desired adjustments.

CreateParams fills in the parameter that is passed as the value of the Params parameter. This data structure holds information needed when telling Windows to create a window handle. The fields of a TCreateParams record become the parameters to a call to the CreateWindowEx API function.

TWinControl implements CreateParams by setting all the fields of Params to generic base values. A number of the standard controls override CreateParams to change one or more of the default values in Params.

See Also

CreateWindowHandle

CreateWnd

AddBiDiModeExStyle

47.1.1.4.12 TOleContainer.DblClick Method

OnDblClick event dispatcher.

Pascal

```
procedure DblClick; override;
```

C++

```
virtual __fastcall DblClick();
```

Description

DblClick calls any event handler attached to the OnDblClick event. Override DblClick to provide other responses in addition to the inherited event-handler call when the user double-clicks the control.

The DblClick method is triggered by left mouse-button double-click message (WM_LBUTTONDBLCLK) from Windows.

See Also

Click

OnDblClick

47.1.1.4.13 TOleContainer.DefineProperties Method

Provides methods to read and write the IsControl property to a stream such as a form file.

Pascal

```
procedure DefineProperties(Filer: TFiler); override;
```

C++

```
virtual __fastcall DefineProperties(TFiler Filer);
```

Description

DefineProperties is called automatically by the streaming system that loads and saves VCL components. It allows persistent objects to read and write properties that are not published. Published properties are loaded and stored automatically.

DefineProperties uses the methods of the filer passed as the Filer parameter to assign methods that can load and save the IsControl property.

Override DefineProperties to persist unpublished properties. Call the DefineBinaryProperty or DefineProperty method of the Filer parameter to assign methods that load or save property values.

Note: When overriding this method, call the parent class's DefineProperties method first.

See Also

DefineBinaryProperty

DefineProperty

DefineProperties

IsControl

47.1.1.4.14 TOleContainer.DestroyObject Method

Destroys the container's OLE object.

Pascal

```
procedure DestroyObject;
```

C++

```
__fastcall DestroyObject();
```

Description

Call DestroyObject to destroy the OLE object, discarding any unsaved changes the user made to it.

See Also

Close (see page 2536)

OleObject (see page 2569)

Creating Clients for Servers That Do Not Have a Type Library

47.1.1.4.15 TOleContainer.DoEnter Method

Respond to receiving input focus.

Pascal

```
procedure DoEnter; override;
```

C++

```
virtual __fastcall DoEnter();
```

Description

DoEnter is called automatically when the control receives the input focus. As implemented in TWinControl, DoEnter calls the OnEnter event handler, if defined.

Descendant classes that override DoEnter should always call the inherited method.

See Also

DoExit

OnEnter

47.1.1.4.16 TOleContainer.DoVerb Method

Requests that the OLE object perform some action.

Pascal

```
procedure DoVerb(Verb: Integer);
```

C++

```
__fastcall DoVerb(int Verb);
```

Description

Call DoVerb to request that the OLE object perform one of the actions listed by the ObjectVerbs (see page 2568) property. OLE defines several verbs, such as ovShow (see page 2626) (to display the OLE object) and ovPrimary (see page 2625) (the default action, usually to activate the OLE object). OLE objects can also define their own custom verbs. Use the ObjectVerbs (see page 2568) property to get a list of those custom verbs.

See Also

ObjectVerbs (see page 2568)

Creating Clients for Servers That Do Not Have a Type Library

47.1.1.4.17 TOleContainer.GetIconMetaPict Method

Returns the global memory handle of a metafile containing the OLE object's icon.

Pascal

```
function GetIconMetaPict: HGlobal;
```

C++

```
__fastcall HGlobal GetIconMetaPict();
```

Description

Call GetIconMetaPict to obtain a metafile handle for drawing the OLE object's icon on another canvas. Use the global

DestroyMetaPict (see page 2617) procedure to destroy the handle. If the OLE container is empty, GetIconMetaPict raises an exception.

See Also

DestroyMetaPict (see page 2617)

TMetafile

TMetafileCanvas

47.1.1.4.18 TOleContainer.GetPopupMenu Method

Returns the pop-up menu associated with the control.

Pascal

```
function GetPopupMenu: TPopupMenu; override;
```

C++

```
virtual __fastcall TPopupMenu GetPopupMenu();
```

Description

Call GetPopupMenu to find the popup menu associated with the control. Override GetPopupMenu to change which popup menu is associated with the control or to specify different popup menus depending on the state of the control.

GetPopupMenu returns the value of the Popup property.

See Also

PopupMenu

47.1.1.4.19 TOleContainer.InsertObjectDialog Method

Displays the Insert Object OLE dialog box.

Pascal

```
function InsertObjectDialog: Boolean;
```

C++

```
__fastcall Boolean InsertObjectDialog();
```

Description

Call InsertObjectDialog to let the user create an OLE object. The OLE object can be embedded or linked. InsertObjectDialog returns true if the dialog box was successfully displayed and the user chose the OK button. InsertObjectDialog returns false otherwise. If the user chose OK and there's already an OLE object in the container, it is destroyed and any changes the user made to it are discarded.

See Also

CreateObjectFromInfo (see page 2539)

ObjectPropertiesDialog (see page 2546)

PasteSpecialDialog (see page 2547)

47.1.1.4.20 TOleContainer.KeyDown Method

Respond to key press events.

Pascal

```
procedure KeyDown(var Key: Word; Shift: TShiftState); override;
```

C++

```
virtual __fastcall KeyDown(Word Key, TShiftState Shift);
```

Description

When a windowed control receives a key-down message (WM_KEYDOWN) from Windows, its message handler calls the DoKeyDown method. If DoKeyDown determines that the control should, in fact, process the character, it calls KeyDown, passing the key code and shift-key state in the Key and Shift parameters, respectively.

KeyDown calls any event handler attached to the OnKeyDown event. Override KeyDown to provide other responses in addition to the event-handler call.

The Key parameter is the key on the keyboard. For non-alphanumeric keys, you must use WinAPI virtual key codes to determine the key pressed. For more information, search for virtual key codes in the Win32 Developer's Reference (WIN32.HLP).

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

Either KeyDown or the OnKeyDown event handler it calls can suppress further processing of a key by setting the Key parameter to zero.

See Also

KeyUp

OnKeyDown

DoKeyDown

DoKeyPress

DoKeyUp

47.1.1.4.21 TOleContainer.LoadFromFile Method

Loads an OLE object from the specified file.

Pascal

```
procedure LoadFromFile(const FileName: string);
```

C++

```
__fastcall LoadFromFile(const UnicodeString FileName);
```

Description

Call LoadFromFile to load an OLE object that was saved using the SaveToFile (see page 2550) method. If OldStreamFormat (see page 2568) is true, LoadFromFile loads files saved by an TOleContainer (see page 2529) object as well as files saved using the current format; if OldStreamFormat (see page 2568) is false, LoadFromFile will not load files saved by the library. If there's already an OLE object in the container, it is destroyed and any changes the user made to it are discarded.

See Also

LoadFromStream (see page 2545)

OldStreamFormat (see page 2568)

SaveToFile (see page 2550)

47.1.1.4.22 TOleContainer.LoadFromStream Method

Loads an OLE object from the specified stream.

Pascal

```
procedure LoadFromStream(Stream: TStream);
```

C++

```
__fastcall LoadFromStream(TStream Stream);
```

Description

Call LoadFromStream to load an OLE object from a stream. If OldStreamFormat (see page 2568) is true, LoadFromStream loads OLE objects saved by a TOleContainer (see page 2529) object as well as OLE objects saved using the current format; if OldStreamFormat (see page 2568) is false, LoadFromStream will not load OLE objects saved by the library. If there's already an OLE object in the container, it is destroyed and any changes the user made to it are discarded.

See Also

LoadFromFile (see page 2544)

OldStreamFormat (see page 2568)

SaveToStream (see page 2550)

TStream

47.1.1.4.23 TOleContainer.MouseDown Method

OnMouseDown event dispatcher.

Pascal

```
procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer);
override;
```

C++

```
virtual __fastcall MouseDown(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user presses the mouse button down while the cursor's hotspot is over the control.

The Button parameter determines which mouse button the user pressed. Shift indicates which shift keys (Shift, Ctrl, or Alt) were down when the user pressed the mouse button. X and Y are the pixel coordinates of the mouse pointer within the client area of the control.

A control calls MouseDown in response to any of the Windows mouse-down messages (WM_LBUTTONDOWN, WM_MBUTTONDOWN, WM_RBUTTONDOWN), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was pressed down: left, right, or middle.

See Also

MouseMove

MouseUp

OnMouseDown

47.1.1.4.24 TOleContainer.ObjectPropertiesDialog Method

Displays the OLE Object Properties property sheet.

Pascal

```
function ObjectPropertiesDialog: Boolean;
```

C++

```
__fastcall Boolean ObjectPropertiesDialog();
```

Description

Call ObjectPropertiesDialog to let the user see and edit the properties of the OLE object. Any changes the user makes in the property sheet are automatically made to the OLE object. ObjectPropertiesDialog returns true if the property sheet was successfully displayed and the user chose the OK button. It returns false otherwise.

If the OLE container is empty, ObjectPropertiesDialog raises an exception.

See Also

DoVerb (see page 2542)

47.1.1.4.25 TOleContainer.Paint Method

Renders the image of a custom control.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method for TCustomControl does nothing other than provide the interface for a method that responds to WM_PAINT messages. Paint is called by the PaintWindow method, after that method has supplied the Canvas with the handle to a device context. When creating a custom control, always override Paint to draw the image of the control.

Tip: To determine which portions of the control's canvas need to be repainted when Paint is called, use the ClipRect property of the canvas.

See Also

Canvas

PaintWindow

GetDeviceContext

ClipRect

47.1.1.4.26 TOleContainer.Paste Method

Pastes the contents of the Windows clipboard as an embedded object.

Pascal

```
procedure Paste;
```

C++

```
__fastcall Paste();
```

Description

Call Paste to embed the object on the clipboard in the OLE container. If there's already an OLE object in the container, it is destroyed and any changes the user made to it are discarded.

See Also

CanPaste (see page 2558)

CreateObjectFromInfo (see page 2539)

PasteSpecialDialog (see page 2547)

47.1.1.4.27 TOleContainer.PasteSpecialDialog Method

Displays the Paste (see page 2546) Special OLE dialog box.

Pascal

```
function PasteSpecialDialog: Boolean;
```

C++

```
__fastcall Boolean PasteSpecialDialog();
```

Description

Call PasteSpecialDialog to let the user control how the contents of the Windows clipboard are pasted into the container. The Paste (see page 2546) Special dialog box lets the user select the format of the data, whether it should be embedded or linked, and whether the OLE object should be displayed as an icon (and if so, to choose a different icon). PasteSpecialDialog returns true if the dialog box was successfully displayed and the user chose the OK button. It returns false otherwise. If the user chose the OK button when there's already an OLE object in the container, it is destroyed and any changes the user made to it are discarded.

See Also

CanPaste (see page 2558)

InsertObjectDialog (see page 2543)

Paste (see page 2546)

47.1.1.4.28 TOleContainer.QueryInterface Method

Returns a reference to a specified interface if the object supports that interface.

Pascal

```
function QueryInterface(const IID: TGUID; out Obj): HResult; override; stdcall;
```

C++

```
virtual __fastcall __stdcall HRESULT QueryInterface(const TGUID IID,  Obj);
```

Description

QueryInterface checks whether the component supports the interface specified by IID, and if so, returns a reference to that interface as the Obj parameter. If the component does not support the interface, the Obj parameter returns nil (Delphi) or NULL (C++).

Note: For components that act as COM object wrapper, QueryInterface calls the QueryInterface method of the internal COM

object.

See Also

_AddRef

_Release

QueryInterface

47.1.1.4.29 TOleContainer.Run Method

Runs the OLE object's server application, but doesn't open or activate the OLE object itself.

Pascal

```
procedure Run;
```

C++

```
__fastcall Run();
```

Description

Call Run to ensure that the server application is running. Once the server application is running, opening and activating the OLE object happens much more quickly. An OLE object must already be loaded in the container before calling Run, Otherwise, Run raises an exception.

See Also

State (see page 2579)

47.1.1.4.30 TOleContainer.SaveAsDocument Method

Saves the OLE object to the specified file in native OLE Document format.

Pascal

```
procedure SaveAsDocument(const FileName: string);
```

C++

```
__fastcall SaveAsDocument(const UnicodeString FileName);
```

Description

Call SaveAsDocument to save the OLE object in its native OLE Document format. An OLE object must already be loaded in the container before calling SaveAsDocument. Otherwise, SaveAsDocument raises an exception.

Note: Saving an OLE object to a file or a stream limits the ability of another application, such as Word, to open the file, even if it is saved with the correct extension. Saving to a file or a stream loses information necessary to open such a file. Use SaveAsDocument when needing to save files in a native OLE Document format.

C++ Examples:

```
/*
This example uses an OLE container and a main menu on a form.
When the user clicks the Open or Save options in the main menu,
an open/save dialog is displayed and by default,
the user is required to enter the name of a Word document.
Files with other extensions supported by the OLE container
can also be accessed. An additional test is made to assure that
the file to open exists and that no file overwriting is
going to take place.
```

```
*/
void __fastcall TForm1::Exit1Click(TObject *Sender)
{
  Close();
}

void __fastcall TForm1::Open1Click(TObject *Sender)
{
  if (OpenDialog1->Execute())
    if (FileExists(OpenDialog1->FileName))
      OleContainer1->CreateObjectFromFile(OpenDialog1->FileName, false);
    else
      throw(Exception("File not found."));
}

void __fastcall TForm1::Save1Click(TObject *Sender)
{
  if (SaveDialog1->Execute())
    if (FileExists(SaveDialog1->FileName))
      throw(Exception("File already exists. Cannot overwrite."));
  else
    OleContainer1->SaveAsDocument(SaveDialog1->FileName);
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  OpenDialog1->Filter = "Documents (*.doc)|*.DOC|Any file (*.*)|*.*";
  SaveDialog1->Filter = "Documents (*.doc)|*.DOC|Any file (*.*)|*.*";
}
```

Delphi Examples:

```
{
This example uses an OLE container and a main menu on a form.
When the user clicks the Open or Save options in the main menu,
an open/save dialog is displayed and by default,
the user is required to enter the name of a Word document.
Files with other extensions supported by the OLE container
can also be accessed. An additional test is made to assure that
the file to open exists and that no file overwriting is
going to take place.
}
procedure TForm1.Exit1Click(Sender: TObject);
begin
  Close;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  OpenDialog1.Filter := 'Documents (*.doc)|*.DOC|Any file (*.*)|*.*';
  SaveDialog1.Filter := 'Documents (*.doc)|*.DOC|Any file (*.*)|*.*';
end;

procedure TForm1.Open1Click(Sender: TObject);
begin
  if OpenDialog1.Execute then
    if FileExists(OpenDialog1.FileName) then
      OleContainer1.CreateObjectFromFile(OpenDialog1.FileName, false)
    else
      raise Exception.Create('File not found.');
end;

procedure TForm1.Save1Click(Sender: TObject);
begin
  if SaveDialog1.Execute then
    if FileExists(SaveDialog1.FileName) then
      raise Exception.Create('File already exists. Cannot overwrite.')
```

```
    else
      OleContainer1.SaveAsDocument(SaveDialog1.FileName);
end;
```

See Also

SaveToFile (see page 2550)

SaveToStream (see page 2550)

47.1.1.4.31 TOleContainer.SaveToFile Method

Saves the OLE object to the specified file.

Pascal

```
procedure SaveToFile(const FileName: string);
```

C++

```
__fastcall SaveToFile(const UnicodeString FileName);
```

Description

Call SaveToFile to save the OLE object. If CopyOnSave (see page 2563) is true, SaveToFile first creates a temporary OLE object that compresses redundant data, saving space in the file that is saved. If OldStreamFormat (see page 2568) is true, SaveToFile writes a file that can be read by a TOleContainer (see page 2529) component.

An OLE object must already be loaded in the container before calling SaveToFile. Otherwise, SaveToFile raises an exception.

See Also

Copy (see page 2536)

CopyOnSave (see page 2563)

LoadFromFile (see page 2544)

OldStreamFormat (see page 2568)

SaveToStream (see page 2550)

47.1.1.4.32 TOleContainer.SaveToStream Method

Saves the OLE object to the specified stream.

Pascal

```
procedure SaveToStream(Stream: TStream);
```

C++

```
__fastcall SaveToStream(TStream Stream);
```

Description

Call SaveToStream to save the OLE object. If CopyOnSave (see page 2563) is true, SaveToStream first creates a temporary OLE object that compresses redundant data, reducing the information that must be saved. If OldStreamFormat (see page 2568) is true, SaveToStream writes the object in a format that can be read by a TOleContainer (see page 2529) component.

An OLE object must already be loaded in the container before calling SaveToStream. Otherwise, SaveToStream raises an exception.

See Also

Copy (see page 2536)

CopyOnSave (see page 2563)

LoadFromStream (see page 2545)

OldStreamFormat (see page 2568)

SaveToFile (see page 2550)

47.1.1.4.33 TOleContainer.UpdateObject Method

Refreshes the OLE object.

Pascal

```
procedure UpdateObject;
```

C++

```
__fastcall UpdateObject();
```

Description

Linked (see page 2567) OLE objects and embedded objects that contain linked OLE objects get out of date when the source of the link is updated. UpdateObject rereads the source to ensure that the OLE object has current data. If there is no OLE object loaded in the container, UpdateObject has no effect.

See Also

Linked (see page 2567)

UpdateVerbs (see page 2551)

47.1.1.4.34 TOleContainer.UpdateVerbs Method

Refreshes the list of verbs the OLE object can respond to.

Pascal

```
procedure UpdateVerbs;
```

C++

```
__fastcall UpdateVerbs();
```

Description

TOleContainer (see page 2529) automatically calls UpdateVerbs when the ObjectVerbs (see page 2568) property is first read, but some OLE objects change their verbs when they perform some actions. The Media Clip OLE object, for example, changes its Play verb to Stop while running. Call UpdateVerbs to ensure that the list of verbs is appropriate to the state of the OLE object.

An OLE object must already be loaded in the container before calling UpdateVerbs. Otherwise, UpdateVerbs raises an exception.

See Also

AutoVerbMenu (see page 2558)

ObjectVerbs (see page 2568)

47.1.1.5 TOleContainer Properties

The properties of the TOleContainer class are listed here.

Public Properties

	Name	Description
R	CanPaste (see page 2558)	Indicates whether the Windows clipboard contains an OLE object that can be pasted into the OLE container.
R	Linked (see page 2567)	Indicates whether the OLE object is linked or embedded.
	Modified (see page 2567)	Indicates whether the OLE object has been modified.
R	NewInserted (see page 2567)	Indicates whether the OLE object was created rather than pasted in or created as a link to a file.
R	ObjectVerbs (see page 2568)	Lists the verbs that the OLE object supports.
R	OleClassName (see page 2569)	Specifies the class name of the OLE object.
R	OleObject (see page 2569)	Represents the OLE object hosted by the OLE container.
R	OleObjectInterface (see page 2569)	Provides access to the IOleObject interface for the OLE object.
R	PrimaryVerb (see page 2576)	Specifies the index of the primary (default) verb for the OLE object.
R	SourceDoc (see page 2579)	Specifies the name of the source document for a linked OLE object.
R	State (see page 2579)	Describes the state of the OLE object.
R	StorageInterface (see page 2579)	Provides access to the IStorage interface for the OLE object.

Published Properties

	Name	Description
	Align (see page 2552)	Determines how the control aligns within its container (parent control).
	AllowActiveDoc (see page 2556)	Specifies whether the OLE container supports the IOleDocumentSite interface.
	AllowInPlace (see page 2556)	Determines whether the OLE container allows in-place activation.
	Anchors (see page 2557)	Specifies how the control is anchored to its parent.
	AutoActivate (see page 2557)	Determines how the object in an OLE container is activated.
	AutoVerbMenu (see page 2558)	Determines whether the OLE container automatically creates a pop-up menu containing the OLE object's verbs.
	BorderStyle (see page 2558)	Determines whether a single line is drawn around the OLE container.
	Caption (see page 2559)	Specifies a text string that identifies the control to the user.
	Color (see page 2561)	Specifies the background color of the control.
	Constraints (see page 2562)	Specifies the size constraints for the control.
	CopyOnSave (see page 2563)	Determines whether the SaveToFile (see page 2550) and SaveToStream (see page 2550) methods first create a copy of the OLE object.
	Ctl3D (see page 2563)	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
	DragCursor (see page 2564)	Indicates the image used to represent the mouse pointer when the control is being dragged.
	DragMode (see page 2564)	Determines how the control initiates drag-and-drop or drag-and-dock operations.
	Enabled (see page 2566)	Controls whether the control responds to mouse, keyboard, and timer events.
	Iconic (see page 2566)	Determines whether the OLE object is displayed as an icon.
	OldStreamFormat (see page 2568)	Indicates whether the OLE object is loaded and saved using the VCL format.
	ParentColor (see page 2570)	Determines where a control looks for its color information.
	ParentCtl3D (see page 2570)	Determines where a component looks to determine if it should appear three dimensional.
	ParentShowHint (see page 2571)	Determines where a control looks to find out if its Help Hint should be shown.
	PopupMenu (see page 2571)	Identifies the pop-up menu associated with the control.
	ShowHint (see page 2577)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	SizeMode (see page 2578)	Determines how the OLE object is sized in the OLE container.
	TabOrder (see page 2580)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 2580)	Determines if the user can tab to a control.
	Visible (see page 2581)	Determines whether the component appears onscreen.

47.1.1.5.1 TOleContainer.Align Property

Determines how the control aligns within its container (parent control).

Pascal

```
property Align;
```

C++

```
__property Align;
```

Description

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes. When the parent is resized, an aligned control also resizes so that it continues to span the top, bottom, left, or right edge of the parent.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel.

Tip: If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

Note: To cause a control to maintain a specified relationship with an edge of its parent, but not necessarily lie along one edge of the parent, use the Anchors property instead.

C++ Examples:

```
/*
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
*/

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  TSplitter* ps = new TSplitter(Form1); // These objects will all be cleaned up by the owner
(Form1).
  TFileListBox* pflb = new TFileListBox(Form1); // The parent (Form1) will clean up these
objects.
  TDirectoryListBox* pdlb = new TDirectoryListBox(Form1);

  // Line up the directory list box on the left of the form.
  pdlb->Parent = Form1;
  pdlb->Align = alLeft;
  pdlb->Width = Form1->ClientWidth/3;

  // Now use the splitter to divide the directory pane from the file pane.
  ps->Parent = Form1;
  ps->Left = pdlb->Left + pdlb->Width + 1; // Move to right of directory list.
  ps->Align = pdlb->Align; // Give same alignment as directory list.

  // Each pane must be at least one quarter of the form's width.
  ps->MinSize = Form1->ClientWidth/4;
```

```
  // Finally, create the last pane - a file list box.
  pflb->Parent = Form1;
  pflb->Align = alClient;
  pdlb->FileList = pflb; // Link the file list box to the directory list box.
}
/*
Dynamically create a TProgressBar control and align it to
the bottom of the form.
*/

#include <ComCtrls.hpp>
#include <Controls.hpp>

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TProgressBar *ProgressBar = new TProgressBar(this);  // The owner will clean this up.
  ProgressBar->Parent = this;
  ProgressBar->Align = alBottom;
  TButton *Button1 = new TButton(this);  // The owner will clean this up.
  Button1->Parent = this;
}
/*
This example dynamically creates a Page Control, then a
series of Tab Sheets on the Page Control.
*/
#include <Comctrls.hpp>
TPageControl* ppc;
const int MAXTABS = 3;
TTabSheet* pts[MAXTABS];
const char * ppcTabTitles[] = { "ShortString", "Orders", "Items", "Parts" };
int iTabTitles = sizeof(ppcTabTitles)/sizeof(ppcTabTitles[0]);
const TColor colorPalette[12] = {
  clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia,
  clTeal, clNavy, clMaroon, clLime, clOlive, clPurple};

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    ppc = new TPageControl(this); // The owner (this) will clean this up.
    ppc->Parent = this;
    ppc->Align = alClient;
    for (int i=0;i<iTabTitles;i++)
  {
        pts[i] = new TTabSheet(this); // The owner (this) will clean this up.
        pts[i]->PageControl = ppc;
        pts[i]->Name = AnsiString("pts") + ppcTabTitles[i];
        pts[i]->Caption = ppcTabTitles[i];
        pts[i]->Brush->Color = colorPalette[i];
  }
}
```

Delphi Examples:

```
{
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  Split: TSplitter;
  Files: TFileListBox;
```

```
  Dirs: TDirectoryListBox;
begin
  { first add a directory list box to the form }
  Dirs := TDirectoryListBox.Create(Form1);
  { line it up on the left of the form }
  Dirs.Parent := Form1;
  Dirs.Align := alLeft;
  Dirs.Width := Form1.ClientWidth div 3;
{ now add the splitter to divide the directory pane from the file pane }
  Split := TSplitter.Create(Form1);
  Split.Parent := Form1;
  { make sure the splitter is to the right of the directory list box! }
  Split.Left := Dirs.Left + Dirs.Width + 1;
  Split.Align := Dirs.Align; { give it the same alignment as the directory }
  { each pane must be at least one quarter of the form's width }
  Split.MinSize := Form1.ClientWidth div 4;
  { Finally, create the last pane - a file list box }
  Files := TFileListBox.Create(Form1);
  Files.Parent := Form1;
  Files.Align := alClient;
  Dirs.FileList := Files; { link the file list box to the directory list box }
end;
{
Dynamically create a TProgressBar control and align it to
the bottom of the form.
}

var
  I: Integer;

procedure TForm1.Button1Click(Sender: TObject);
var
  ProgressBar: TProgressBar;
begin
  ProgressBar := TProgressBar.Create(Self);
  with ProgressBar do
  begin
    Parent := Self;
    Align := alBottom;
  end;
end;
{
This example dynamically creates a Page Control, then a
series of Tab Sheets on the Page Control.
}
var
  PageControl1: TPageControl;
  pts: array[0..3] of TTabSheet;
const
  TabTitles: array[0..3] of ShortString = ('Customer', 'Orders', 'Items', 'Parts' );

procedure TForm1.FormCreate(Sender: TObject);
const
  colorpalette: Array[0..11] of TColor = (
    clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia, clTeal,
    clNavy, clMaroon, clLime, clOlive, clPurple);
var
  i: Integer;
begin
  PageControl1 := TPageControl.Create(Self);
  PageControl1.Parent := Self;
  PageControl1.Align := alClient;
  for i := Low(TabTitles) to High(TabTitles) do
  begin
    pts[i]:= TTabSheet.Create(PageControl1);
    with pts[i] do
```

```
      begin
        PageControl := PageControl1;
        Name := 'ts' + TabTitles[i];
        Caption := TabTitles[i];
        Brush.Color := colorPalette[i];
     end;
    end;
  end;

  procedure TForm1.FormDestroy(Sender: TObject);
  var
    i: Integer;
  begin
    for i := Low(TabTitles) to High(TabTitles) do
    begin
      pts[i].Free;
    end;
    PageControl1.Free;
  end;
```

See Also

Parent

Anchors

47.1.1.5.2 TOleContainer.AllowActiveDoc Property

Specifies whether the OLE container supports the IOleDocumentSite interface.

Pascal

```
property AllowActiveDoc: Boolean;
```

C++

```
__property Boolean AllowActiveDoc;
```

Description

Set AllowActiveDoc to true to specify that the OLE container can house an ActiveDoc (DocObject). Set AllowActiveDoc to false to limit the OLE container to hosting OLE controls that use the more usual interfaces such as IOleClientSite. ActiveDoc-enabled controls have a slightly different border. Set AllowActiveDoc to false to maintain a consistent appearance in forms that do not need to support the IOleDocumentSite interface.

See Also

AllowInPlace (see page 2556)

47.1.1.5.3 TOleContainer.AllowInPlace Property

Determines whether the OLE container allows in-place activation.

Pascal

```
property AllowInPlace: Boolean;
```

C++

```
__property Boolean AllowInPlace;
```

Description

Set AllowInPlace to specify what happens when the user activates the OLE object. If AllowInPlace is true and Iconic (see page 2566) is false, the OLE object is activated in place. If AllowInPlace is false, the OLE object is activated in a separate window. The default value is true.

See Also

AllowActiveDoc (see page 2556)

AutoActivate (see page 2557)

Iconic (see page 2566)

47.1.1.5.4 TOleContainer.Anchors Property

Specifies how the control is anchored to its parent.

Pascal

```
property Anchors;
```

C++

```
__property Anchors;
```

Description

Use Anchors to ensure that a control maintains its current position relative to an edge of its parent, even if the parent is resized. When its parent is resized, the control holds its position relative to the edges to which it is anchored.

If a control is anchored to opposite edges of its parent, the control stretches when its parent is resized. For example, if a control has its Anchors property set to [akLeft, akRight], the control stretches when the width of its parent changes.

Anchors is enforced only when the parent is resized. Thus, for example, if a control is anchored to opposite edges of a form at design time and the form is created in a maximized state, the control is not stretched because the form is not resized after the control is created.

Note: If a control should maintain contact with three edges of its parent (hugging one side of the parent and stretching the length of that side), use the Align property instead. Unlike Anchors, Align allows controls to adjust to changes in the size of other aligned sibling controls as well as changes to the parent's size.

See Also

Parent

Align

47.1.1.5.5 TOleContainer.AutoActivate Property

Determines how the object in an OLE container is activated.

Pascal

```
property AutoActivate: TAutoActivate;
```

C++

```
__property TAutoActivate AutoActivate;
```

Description

Set AutoActivate to specify what actions activate the OLE object. AutoActivate takes one of the values in the following table:

Values	Meaning
aaManual	The OLE object must be activated programmatically by calling DoVerb (see page 2542) (ovShow (see page 2626)).
aaGetFocus	The OLE object is activated when the OLE container gets focus (by clicking on it with the mouse or pressing Tab to move the focus to it).

aaDoubleClick	(Default) The OLE object is activated by double-clicking it or pressing Enter while the container has the focus.

See Also

AllowActiveDoc (see page 2556)

DoVerb (see page 2542)

47.1.1.5.6 TOleContainer.AutoVerbMenu Property

Determines whether the OLE container automatically creates a pop-up menu containing the OLE object's verbs.

Pascal

```
property AutoVerbMenu: Boolean;
```

C++

```
__property Boolean AutoVerbMenu;
```

Description

Use AutoVerbMenu to specify whether the OLE object can replace the OLE container's popup menu. If AutoVerbMenu is true (the default), the OLE object replaces any existing PopupMenu (see page 2571) property. If AutoVerbMenu is false, no popup menu is automatically created.

See Also

DoVerb (see page 2542)

ObjectVerbs (see page 2568)

PopupMenu

47.1.1.5.7 TOleContainer.BorderStyle Property

Determines whether a single line is drawn around the OLE container.

Pascal

```
property BorderStyle: TBorderStyle;
```

C++

```
__property TBorderStyle BorderStyle;
```

Description

Set BorderStyle to specify whether the OLE container should be framed by a single line. These are the possible values:

Value	Meaning
bsNone	No visible border
bsSingle	Single-line border

See Also

Ctl3D

47.1.1.5.8 TOleContainer.CanPaste Property

Indicates whether the Windows clipboard contains an OLE object that can be pasted into the OLE container.

Pascal

```
property CanPaste: Boolean;
```

C++

```
__property Boolean CanPaste;
```

Description

Read CanPaste before trying to paste or paste link an OLE object from the clipboard.

See Also

Paste (see page 2546)

PasteSpecialDialog (see page 2547)

47.1.1.5.9 TOleContainer.Caption Property

Specifies a text string that identifies the control to the user.

Pascal

```
property Caption;
```

C++

```
__property Caption;
```

Description

Use Caption to specify the text string that labels the control.

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

Note: Controls that display text use either the Caption property or the Text property to specify the text value. Which property is used depends on the type of control. In general, Caption is used for text that appears as a window title or label, while Text is used for text that appears as the content of a control.

Note: The Caption property inheritted from TControl is a protected property. Access to the Caption requires casting the class to an inheritted one where Caption is a published or public property.

C++ Examples:

```
/*
The following code keeps the size of the label control
constant, even though the length of the label's caption
changes. As a result, the caption of the label is probably
too long to display in the label when the user clicks the
button:
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Label1->AutoSize = false;
  Label1->Caption = "This string is too long to be the caption of this label";
}
/*
This example requires a TPageControl populated by several
TabSheets, a TEdit and a TButton. When the button is
clicked, the caption of the active tab changes to that of
```

```
the TEdit.  These lines of code could also be written:
PageControl1.ActivePage.Caption := Edit1.Text;
This would make a TTabSheet variable unnecessary since
PageControl1.->ActivePage is already of type TTabSheet.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TTabSheet *sheet = PageControl1->ActivePage;
  sheet->Caption = Edit1->Text;
}
/*
This example creates a ToolBar and adds ToolButtons to it.
Include ComCtlrs in the uses clause.
*/
#include <comctrls.hpp>
#include <memory>

void AddButtons(TToolBar *toolBar, TStringList *captions)
{
  for (int i = 0; i < captions->Count; i++)
  {
    TToolButton *button = new TToolButton(toolBar);  // The owner (toolBar) will clean this up.
    button->Parent = toolBar;
    button->Caption = captions->Strings[i];
    if (button->Caption == "|")
      button->Style = tbsSeparator;
    else
      button->Style = tbsButton;
  }
}

TToolBar *myToolBar;

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  myToolBar = new TToolBar(this); // The owner (this) will clean this up.
  myToolBar->Parent = this;
//  TStringList *captions = new TStringList();
  std::auto_ptr<TStringList> captions(new TStringList());

  captions->Add("New");
  captions->Add("Save");
  captions->Add("|");
  captions->Add("Cut");
  captions->Add("Copy");
  captions->Add("Paste");
  AddButtons(myToolBar, captions.get());
//  delete captions;
  myToolBar->ShowCaptions = True;
  myToolBar->Height = 40;
}
```

Delphi Examples:

```
{
The following code keeps the size of the label control
constant, even though the length of the label's caption
changes. As a result, the caption of the label is probably
too long to display in the label when the user clicks the
button:
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.AutoSize := False;
  Label1.Caption :=
    'This string is too long to be the caption of this label';
end;
```

```
{
This example requires a TPageControl populated by several
TabSheets, a TEdit and a TButton. When the button is
clicked, the caption of the active tab changes to that of
the TEdit.  These lines of code could also be written:
PageControl1.ActivePage.Caption := Edit1.Text;
This would obviate the need for a TTabSheet variable since
PageControl1.->ActivePage is already of type TTabSheet.
}
procedure TForm1.Button1Click(Sender: TObject);
var
  sheet: TTabSheet;
begin
  sheet := PageControl1.ActivePage;
  sheet.Caption := Edit1.Text;
end;
{
This example creates a ToolBar and adds ToolButtons to it.
Include ComCtlrs in the uses clause.
}
procedure AddButtons(ToolBar: TToolBar;
  Const ButtonCaptions: array of String);
var
  I: Integer;
begin
  for I := 0 to High(ButtonCaptions) do
    begin
      with TToolButton.Create(ToolBar) do
      begin
        Parent := ToolBar;
        Caption := ButtonCaptions[I];
        if (ButtonCaptions[I] = '|') then
          Style := tbsSeparator
        else
          Style := tbsButton;
      end;
    end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1 := TToolBar.Create(Self);
  ToolBar1.Parent := Self;
  AddButtons(ToolBar1, ['New', 'Save', '|', 'Cut', 'Copy', 'Paste']);
  ToolBar1.ShowCaptions := True;
  ToolBar1.Height := 40;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  I: Integer;
begin
  for I := ToolBar1.ButtonCount - 1 downto 0 do
    ToolBar1.Buttons[I].Free;
  ToolBar1.Free;
end;
```

47.1.1.5.10 TOleContainer.Color Property

Specifies the background color of the control.

Pascal

```
property Color;
```

C++

```
__property Color;
```

Description

Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to false.

C++ Examples:

```
/*
The following code changes the color of a shape control
using the Color dialog box. The example displays the Color
dialog box when the Button1 button is clicked, allowing the
user to select a color with the dialog box. When a color
value is selected with the dialog box, it is assigned to
the shape control.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  if (ColorDialog1->Execute())
    Shape1->Brush->Color = ColorDialog1->Color;
}
```

Delphi Examples:

```
{
The following code changes the color of a shape control
using the Color dialog box. The example displays the Color
dialog box when the Button1 button is clicked, allowing the
user to select a color with the dialog box. When a color
value is selected with the dialog box, it is assigned to
the shape control.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  if ColorDialog1.Execute then
    Shape1.Brush.Color := ColorDialog1.Color;
end;
```

See Also

ParentColor

47.1.1.5.11 TOleContainer.Constraints Property

Specifies the size constraints for the control.

Pascal

```
property Constraints;
```

C++

```
__property Constraints;
```

Description

Use Constraints to specify the minimum and maximum width and height of the control. When Constraints contains maximum or minimum values, the control cannot be resized to violate those constraints.

Note: Constraints apply to the height of the form. However, the height of the form depends on how large Windows makes the title bar. For example, Windows XP uses much larger title bars than other windowing systems. To work around this, note the

ClientHeight when you design your form and set the constraints in the FormCreate event. In the following example, ClientHeight is represented by x: `Constraints.MinHeight := x + Height - ClientHeight`.

Warning: Do not set up constraints that conflict with the value of the Align or Anchors property. When these properties conflict, the response of the control to resize attempts is not well-defined.

See Also

AutoSize

Align

Anchors

OnConstrainedResize

OnCanResize

47.1.1.5.12 TOleContainer.CopyOnSave Property

Determines whether the SaveToFile (see page 2550) and SaveToStream (see page 2550) methods first create a copy of the OLE object.

Pascal

```
property CopyOnSave: Boolean;
```

C++

```
__property Boolean CopyOnSave;
```

Description

When CopyOnSave is true, the SaveToFile (see page 2550) and SaveToStream (see page 2550) methods create a temporary OLE object, compressing redundant data to save space. This compressed copy is then saved to the file or stream. When CopyOnSave is false, the OLE object is saved directly to the file or stream without using a temporary copy to compress redundant information.

Set CopyOnSave to false when the container holds very large embedded objects and there might not be enough memory to make the temporary copy. The default value is true.

See Also

OleObject (see page 2569)

SaveToFile (see page 2550)

SaveToStream (see page 2550)

47.1.1.5.13 TOleContainer.Ctl3D Property

Determines whether a control has a three-dimensional (3-D) or two-dimensional look.

Pascal

```
property Ctl3D;
```

C++

```
__property Ctl3D;
```

Description

Ctl3D is provided for backward compatibility. It is not used by 32-bit versions of Windows or NT4.0 and later, with the exception of Windows XP Home, where it is used on occasion.

On earlier platforms, Ctl3D controlled whether the control had a flat or beveled appearance.

See Also

ParentCtl3D

BevelKind

BevelInner

BevelOuter

BorderWidth

47.1.1.5.14 TOleContainer.DragCursor Property

Indicates the image used to represent the mouse pointer when the control is being dragged.

Pascal

```
property DragCursor;
```

C++

```
__property DragCursor;
```

Description

Use the DragCursor property to change the cursor image presented when the control is being dragged.

Note: To make a custom cursor available for the DragCursor property, see the Cursor property.

See Also

Cursor

DragKind

DragMode

47.1.1.5.15 TOleContainer.DragMode Property

Determines how the control initiates drag-and-drop or drag-and-dock operations.

Pascal

```
property DragMode;
```

C++

```
__property DragMode;
```

Description

Use DragMode to control when the user can drag the control. Disable the drag-and-drop or drag-and-dock capability at runtime by setting the DragMode property value to dmManual. Enable automatic dragging by setting DragMode to dmAutomatic.

C++ Examples:

```
/*
The following code uses GetNodeAt to add a dragged node as a
child of the node under the mouse when it is dropped.  This
```

```
example requires a populated TreeView.  Also, the TTreeView
DragMode property set to dmAutomatic, and the TTreeView
OnDragOver event handler must be implemented to accept the
drop.
*/
void __fastcall TForm1::TreeView1DragDrop(TObject *Sender, TObject *Source,
      int X, int Y)
{
  if (Source != dynamic_cast<TObject *>(TreeView1) || TreeView1->Selected == NULL)
    return;
  THitTests HT = TreeView1->GetHitTestInfoAt(X, Y);
  TNodeAttachMode AttachMode;
  TTreeNode *Item = TreeView1->GetNodeAt(X, Y);
  if (HT.Contains(htOnItem) || HT.Contains(htOnIcon))
    AttachMode = naAddChild;
  else if (HT.Contains(htNowhere))
    AttachMode = naAdd;
  else if (HT.Contains(htOnIndent))
    AttachMode = naInsert;
  else
    return;
  TreeView1->Selected->MoveTo(Item, AttachMode);
}

void __fastcall TForm1::TreeView1DragOver(TObject *Sender, TObject *Source,
      int X, int Y, TDragState State, bool &Accept)
{
  Accept = (Source == dynamic_cast<TObject *>(TreeView1));
}
```

Delphi Examples:

```
{
The following code uses GetNodeAt to add a dragged node as a
child of the node under the mouse when it is dropped.  This
example requires a populated TreeView.  Also, the TTreeView
DragMode property must be set to dmAutomatic, and the TTreeView
OnDragOver event handler must be implemented to accept the
drop.
}
procedure TForm1.TreeView1DragDrop(Sender, Source: TObject; X, Y: Integer);
var
  AnItem: TTreeNode;
  AttachMode: TNodeAttachMode;
  HT: THitTests;
begin
  if TreeView1.Selected = nil then Exit;
  HT := TreeView1.GetHitTestInfoAt(X, Y);
  AnItem := TreeView1.GetNodeAt(X, Y);
  if (HT - [htOnItem, htOnIcon, htNowhere, htOnIndent] <> HT) then
  begin
    if (htOnItem in HT) or (htOnIcon in HT) then
      AttachMode := naAddChild
    else if htNowhere in HT then AttachMode := naAdd
    else if htOnIndent in HT then AttachMode := naInsert;
    TreeView1.Selected.MoveTo(AnItem, AttachMode);
  end;
end;

procedure TForm1.TreeView1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TTreeView;
end;
```

See Also

BeginDrag

DragCursor

EndDrag

OnDragDrop

OnDragOver

OnEndDrag

OnStartDrag

DragKind

Arranging the Order of Persistent Columns

47.1.1.5.16 TOleContainer.Enabled Property

Controls whether the control responds to mouse, keyboard, and timer events.

Pascal

```
property Enabled;
```

C++

```
__property Enabled;
```

Description

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to False. Some disabled controls appear dimmed (for example: buttons, check boxes, labels), while others (container controls) simply lose their functionality without changing their appearance. If Enabled is set to False, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to True.

Note: This property applies to all TControl descendants.

See Also

Visible

DisableControls

EnableControls

Enabling Mouse

47.1.1.5.17 TOleContainer.Iconic Property

Determines whether the OLE object is displayed as an icon.

Pascal

```
property Iconic: Boolean;
```

C++

```
__property Boolean Iconic;
```

Description

Set Iconic to specify whether the OLE container displays the icon for the server application instead of the image of the object. If Iconic is true, the OLE object is displayed as an icon; if false, it's displayed as it would be in the server application.

See Also

GetIconMetaPict (see page 2542)

47.1.1.5.18 TOleContainer.Linked Property

Indicates whether the OLE object is linked or embedded.

Pascal

```
property Linked: Boolean;
```

C++

```
__property Boolean Linked;
```

Description

Read Linked to determine whether the OLE object is linked to the OLE server. If Linked is true, the OLE object is linked; if false, it's embedded. An OLE object must already be loaded in the container before accessing the Linked property.

See Also

UpdateObject (see page 2551)

47.1.1.5.19 TOleContainer.Modified Property

Indicates whether the OLE object has been modified.

Pascal

```
property Modified: Boolean;
```

C++

```
__property Boolean Modified;
```

Description

Use Modified to determine when the OLE object has been modified (including being deleted or replaced by another OLE object). To keep track of incremental changes, set Modified to false and later check it to see if the OLE object has been modified after since Modified was set to false. If there is no OLE object loaded into the container, Modified returns false.

See Also

OleObject (see page 2569)

47.1.1.5.20 TOleContainer.NewInserted Property

Indicates whether the OLE object was created rather than pasted in or created as a link to a file.

Pascal

```
property NewInserted: Boolean;
```

C++

```
__property Boolean NewInserted;
```

Description

Read NewInserted to determine whether the OLE object was newly created via a call to InsertObjectDialog (see page 2543). If there is no OLE object loaded into the container, NewInserted returns false. If NewInserted is true after a call to InsertObjectDialog (see page 2543), call DoVerb (see page 2542)(ovShow (see page 2626)) to let the user edit the object.

See Also

DoVerb (see page 2542)

InsertObjectDialog (see page 2543)

OleObject (see page 2569)

47.1.1.5.21 TOleContainer.ObjectVerbs Property

Lists the verbs that the OLE object supports.

Pascal

```
property ObjectVerbs: TStrings;
```

C++

```
__property TStrings ObjectVerbs;
```

Description

Read ObjectVerbs to determine the verbs that can be used with the OLE object. The verb names can have embedded ampersand (&) characters to indicate shortcut keys. An OLE object must already be loaded in the container before accessing the ObjectVerbs property.

See Also

AutoVerbMenu (see page 2558)

DoVerb (see page 2542)

PrimaryVerb (see page 2576)

TStrings

47.1.1.5.22 TOleContainer.OldStreamFormat Property

Indicates whether the OLE object is loaded and saved using the VCL format.

Pascal

```
property OldStreamFormat: Boolean;
```

C++

```
__property Boolean OldStreamFormat;
```

Description

Set OldStreamFormat to true to read OLE objects that were saved to file using VCL or to save OLE objects that will be used by VCL. If OldStreamFormat is false (the default), the LoadFromFile (see page 2544), LoadFromStream (see page 2545), SaveToFile (see page 2550), and SaveToStream (see page 2550) methods use the newer format that contains extra header information. The extra header information is used to validate a file or stream. This property is provided for backward compatibility only.

See Also

LoadFromFile (see page 2544)

LoadFromStream (see page 2545)

SaveToFile (see page 2550)

SaveToStream (see page 2550)

47.1.1.5.23 TOleContainer.OleClassName Property

Specifies the class name of the OLE object.

Pascal

```
property OleClassName: string;
```

C++

```
__property UnicodeString OleClassName;
```

Description

Read OleClassName to obtain the class name (programmatic identifier) for the OLE object. The class name is a string describing the class of OLE object that is registered in the Windows registry. An OLE object must already be loaded in the container before accessing the OleClassName property.

See Also

OleObject (see page 2569)

47.1.1.5.24 TOleContainer.OleObject Property

Represents the OLE object hosted by the OLE container.

Pascal

```
property OleObject: Variant;
```

C++

```
__property Variant OleObject;
```

Description

Read OleObject to access the OLE object hosted by the OLE container as a variant. To interact with the OLE object using the IOleObject interface, use the OleObjectInterface (see page 2569) property instead. If the OLE object is not loaded in the container before reading OleObject, an EOleError exception is raised.

See Also

EOleError

OleObjectInterface (see page 2569)

47.1.1.5.25 TOleContainer.OleObjectInterface Property

Provides access to the IOleObject interface for the OLE object.

Pascal

```
property OleObjectInterface: IOleObject;
```

C++

```
__property IOleObject OleObjectInterface;
```

Description

Use OleObjectInterface for low-level access to the OLE API. If there is no OLE object loaded into the container, OleObjectInterface returns nil (Delphi) or NULL (C++).

See Also

OleObject (see page 2569)

Creating Clients for Servers That Do Not Have a Type Library

47.1.1.5.26 TOleContainer.ParentColor Property

Determines where a control looks for its color information.

Pascal

```
property ParentColor;
```

C++

```
__property ParentColor;
```

Description

To have a control use the same color as its parent control, set ParentColor to true. If ParentColor is false, the control uses its own Color property.

Set ParentColor to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentColor is true for all controls in a form, changing the background color of the form to gray causes all the controls on the form to also have a gray background.

When the value of a control's Color property changes, ParentColor becomes false automatically.

See Also

Color

Parent

ParentFont

ParentShowHint

47.1.1.5.27 TOleContainer.ParentCtl3D Property

Determines where a component looks to determine if it should appear three dimensional.

Pascal

```
property ParentCtl3D;
```

C++

```
__property ParentCtl3D;
```

Description

ParentCtl3D is provided for backwards compatibility. It has no effect on 32-bit versions of Windows or NT 4.0 and later.

ParentCtl3D determines whether the control uses its parent's Ctl3D property.

See Also

Ctl3D

47.1.1.5.28 TOleContainer.ParentShowHint Property

Determines where a control looks to find out if its Help Hint should be shown.

Pascal

```
property ParentShowHint;
```

C++

```
__property ParentShowHint;
```

Description

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the control uses the ShowHint property value of its parent. If ParentShowHint is false, the control uses the value of its own ShowHint property.

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note: Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See Also

Hint

Parent

ParentColor

ParentFont

ShowHint

47.1.1.5.29 TOleContainer.PopupMenu Property

Identifies the pop-up menu associated with the control.

Pascal

```
property PopupMenu;
```

C++

```
__property PopupMenu;
```

Description

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the control and clicks the right mouse button. If the TPopupMenu's AutoPopup property is true, the pop-up menu appears automatically. If the menu's AutoPopup property is false, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

C++ Examples:

```
/*
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  PopupMenu1->AutoPopup = true;
  Edit1->PopupMenu = PopupMenu1;
```

```
  Edit2->PopupMenu = PopupMenu1;
}

/*
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
*/
void __fastcall TForm1::Copy1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CopyToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Cut1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CutToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Paste1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->PasteFromClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}
/*
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
```

```
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
    {
        switch (ClickedItem->Tag)
        {
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
            }
        }
    }
}

/*
To edit or destroy an item, grab its pointer via the Items
property.
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}

void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
```

```
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  PopupMenu1.AutoPopup := True;
  Edit1.PopupMenu := PopupMenu1;
  Edit2.PopupMenu := PopupMenu1;
end;

{
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
}
procedure TForm1.Copy1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
   Edit1.CopyToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CopyToClipboard
  else
    Beep;
end;

procedure TForm1.Cut1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.CutToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CutToClipboard
  else
    Beep;
end;

procedure TForm1.Paste1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.PasteFromClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.PasteFromClipboard
  else
    Beep;
end;
{
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
```

```
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
}

type
  TForm1 = class(TForm)
    AddButton: TButton;
    EditButton: TButton;
    DestroyButton: TButton;
    PopupMenu1: TPopupMenu;
    procedure AddButtonClick(Sender: TObject);
    procedure EditButtonClick(Sender: TObject);
    procedure DestroyButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    procedure PopupMenuItemsClick(Sender: TObject);

  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.AddButtonClick(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
    // The owner (PopupMenu1) will cleanup this menu item.
  NewItem := TMenuItem.Create(PopupMenu1); // create the new item
  index := PopupMenu1.Items.Count;
  PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
  NewItem.Caption := 'Menu Item ' + IntToStr(index);
  NewItem.Tag := index;
  NewItem.OnClick :=
    PopupMenuItemsClick; // assign it an event handler
end;

procedure TForm1.PopupMenuItemsClick(Sender: TObject);
begin
  with Sender as TMenuItem do
  begin
    case Tag of
      0:  ShowMessage('first item clicked');
      1:  ShowMessage('second item clicked');
      2:  ShowMessage('third item clicked');
      3:  ShowMessage('fourth item clicked');
    end;
  end;
end;

{
To edit or destroy an item, grab its pointer via the Items
property.
}
procedure TForm1.EditButtonClick(Sender: TObject);
var
  ItemToEdit: TMenuItem;
```

```
begin
  ItemToEdit := PopupMenu.Items[1];
  ItemToEdit.Caption := 'Changed Caption';
end;

procedure TForm1.DestroyButtonClick(Sender: TObject);
var
  ItemToDelete: TMenuItem;
begin
  ItemToDelete := PopupMenu.Items[2];
  ItemToDelete.Free;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
  for index := 0 to 3 do
  begin
      // The owner (PopupMenu1) will cleanup this menu item.
    NewItem := TMenuItem.Create(PopupMenu1); // create the new item
    PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
    NewItem.Caption := 'Menu Item ' + IntToStr(index);
    NewItem.Tag := index;
    NewItem.OnClick :=
      PopupMenuItemsClick; // assign it an event handler
  end;
end;
```

See Also

AutoPopup (see page 1437)

Popup (see page 1434)

OnContextPopup

47.1.1.5.30 TOleContainer.PrimaryVerb Property

Specifies the index of the primary (default) verb for the OLE object.

Pascal

```
property PrimaryVerb: Integer;
```

C++

```
__property int PrimaryVerb;
```

Description

Use PrimaryVerb to determine which verb in the ObjectVerbs (see page 2568) list specifies the default action for the OLE object. In a call to the DoVerb (see page 2542) method, DoVerb (see page 2542)(ovPrimary (see page 2625)) is the same as DoVerb (see page 2542)(PrimaryVerb). An OLE object must already be loaded in the container before accessing the PrimaryVerb property.

See Also

DoVerb (see page 2542)

ObjectVerbs (see page 2568)

47.1.1.5.31 TOleContainer.ShowHint Property

Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.

Pascal

```
property ShowHint;
```

C++

```
__property ShowHint;
```

Description

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either

the control's own ShowHint property must be true, or

the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

C++ Examples:

```
/*
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  Edit1->Hint = "Enter your name";
  Edit1->ShowHint = true;

  char string[10];
  char index[3];
  for(int i = 1; i <= 10; i++)
  {
    strcpy(string, "Item");
    itoa(i, index, 10);
    strcat(string, index);
    ListBox1->Items->Add(string);
  }
  ListBox1->Hint = "Select an item";
  ListBox1->ShowHint = true;
  Application->ShowHint = true;
}
/*
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
*/
```

Delphi Examples:

```
{
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
```

```
}
procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
  Edit1.Hint := 'Enter your name';
  Edit1.ShowHint := True;
  with ListBox1 do
  begin
    for I := 1 to 10 do
      Items.Add('Item ' + IntToStr(I));
    Hint := 'Select an item';
    ShowHint := True;
  end;
  Application.ShowHint := True;
end;
{
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
}
```

See Also

Hint

ParentShowHint

ShowHint

Displaying Fly-over Help

47.1.1.5.32 TOleContainer.SizeMode Property

Determines how the OLE object is sized in the OLE container.

Pascal

```
property SizeMode: TSizeMode;
```

C++

```
__property TSizeMode SizeMode;
```

Description

Set SizeMode to specify how the OLE object is fit into the client area of the OLE container. The following table gives the possible values :

Value	Meaning
smClip	(Default)- Displays the OLE object at its normal size, clipping any parts that don't fit within the container.
smCenter	Displays the OLE object at its normal size, centering it within the container.
smScale	Scales or shrinks the view of the OLE object to fit within the container, by scaling width and height proportionally.
smStretch	Scales or shrinks the view of the OLE object to fill the OLE container, without regard to preserving the proportions of the OLE object.
smAutoSize	Displays the OLE object at its normal size and automatically resizes the container to fit the size of the OLE object.

See Also

AllowInPlace (see page 2556)

ClientRect

47.1.1.5.33 TOleContainer.SourceDoc Property

Specifies the name of the source document for a linked OLE object.

Pascal

```
property SourceDoc: string;
```

C++

```
__property UnicodeString SourceDoc;
```

Description

Read SourceDoc to obtain the name of the OLE object when it is linked. An OLE object must already be loaded in the container before accessing the SourceDoc property. If the OLE object isn't linked, SourceDoc returns an empty string.

See Also

Linked (see page 2567)

OleClassName (see page 2569)

47.1.1.5.34 TOleContainer.State Property

Describes the state of the OLE object.

Pascal

```
property State: TObjectState;
```

C++

```
__property TObjectState State;
```

Description

Read State to determine the state of the OLE object and its server application. State has one of the following values:

Values	Meaning
osEmpty	There is no OLE object in the container.
osLoaded	There is an OLE object in the container, but its server application isn't currently running.
osRunning	The OLE object's server is running.
osOpen	The OLE object is open in a separate window.
osInPlaceActive	The OLE object is activated in place, but hasn't yet merged its menus or toolbars. This is an intermediate state; the State property will shift to osUIActive as soon as the menus and toolbars have been merged.
osUIActive	The OLE object is activated in place and menus and toolbars have been merged

See Also

AllowInPlace (see page 2556)

Iconic (see page 2566)

47.1.1.5.35 TOleContainer.StorageInterface Property

Provides access to the IStorage interface for the OLE object.

Pascal

```
property StorageInterface: IStorage;
```

C++

```
__property IStorage StorageInterface;
```

Description

Use StorageInterface for low-level access to the OLE API. If there is no OLE object loaded into the container, StorageInterface returns nil (Delphi) or NULL (C++).

See Also

OleObjectInterface (see page 2569)

47.1.1.5.36 TOleContainer.TabOrder Property

Indicates the position of the control in its parent's tab order.

Pascal

```
property TabOrder;
```

C++

```
__property TabOrder;
```

Description

TabOrder is the order in which child windows are visited when the user presses the Tab key. The control with the TabOrder value of 0 is the control that has the focus when the form first appears.

Initially, the tab order is always the order in which the controls were added to the form. The first control added to the form has a TabOrder value of 0, the second is 1, the third is 2, and so on. Change this by changing the TabOrder property.

Each control has a unique tab-order value within its parent. If you change the TabOrder property value of one control to be the same as the value of a second control, the TabOrder value for all the other controls changes. For example, suppose a control is sixth in the tab order. If you change the control's TabOrder property value to 3 (making the control fourth in the tab order), the control that was originally fourth in the tab order now becomes fifth, and the control that was fifth becomes sixth.

Assigning TabOrder a value greater than the number of controls contained in the parent control moves the control to the end of the tab order. The control does not take on the assigned value of TabOrder, but instead is given the number that assures the control is the last in the tab order.

Note: TabOrder is meaningful only if the TabStop property is true and if the control has a parent. (The TabOrder property of a form is not used unless the form is the child of another form.) A control with a TabOrder of -1 has no parent, and therefore cannot be reached by pressing the Tab key. To remove a parented control from the Tab order, set its TabStop property to false.

See Also

FindNextControl

TabStop

47.1.1.5.37 TOleContainer.TabStop Property

Determines if the user can tab to a control.

Pascal

```
property TabStop;
```

C++

```
__property TabStop;
```

Description

Use the TabStop to allow or disallow access to the control using the Tab key.

If TabStop is true, the control is in the tab order. If TabStop is false, the control is not in the tab order and the user can't press the Tab key to move to the control.

Note: TabStop is not meaningful for a form unless the form assigns another form to be its parent.

See Also

TabOrder

47.1.1.5.38 TOleContainer.Visible Property

Determines whether the component appears onscreen.

Pascal

```
property Visible;
```

C++

```
__property Visible;
```

Description

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

Calling the Show method sets the control's Visible property to true. Calling the Hide method sets it to false.

C++ Examples:

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
    pPage->PageControl = PageControl1;
    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this); // The owner will clean this up.
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
```

```
  PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
    begin
      PageControl := PageControl1;
      Caption := 'TabSheet' + IntToStr(i);
      with TEdit.Create(Self) do
      begin
        Parent := PageControl1.Pages[i];
        Left := Random(PageControl1.ActivePage.ClientWidth - Width);
        Top := Random(PageControl1.ActivePage.ClientHeight - Height);
      end;
    end;
    PageControl1Change(Sender);
end;
```

See Also

Enabled

Hide

Show

VisibleChanging

47.1.1.6 TOleContainer Events

The events of the TOleContainer class are listed here.

Published Events

	Name	Description
	OnActivate (see page 2583)	Occurs immediately after the OLE object is activated.
	OnDeactivate (see page 2583)	Occurs immediately after the OLE object is deactivated.
	OnDragDrop (see page 2584)	Occurs when the user drops an object being dragged.
	OnDragOver (see page 2587)	Occurs when the user drags an object over a control.
	OnEndDrag (see page 2590)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.

	OnEnter (see page 2591)	Occurs when a control receives the input focus.
	OnExit (see page 2593)	Occurs when the input focus shifts away from one control to another.
	OnKeyDown (see page 2597)	Occurs when a user presses any key while the control has focus.
	OnKeyPress (see page 2601)	Occurs when a key is pressed.
	OnKeyUp (see page 2602)	Occurs when the user releases a key that has been pressed.
	OnMouseActivate (see page 2604)	Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.
	OnMouseDown (see page 2605)	Occurs when the user presses a mouse button with the mouse pointer over a control.
	OnMouseMove (see page 2607)	Occurs when the user moves the mouse pointer while the mouse pointer is over a control.
	OnMouseUp (see page 2609)	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.
	OnObjectMove (see page 2611)	Occurs when the OLE container is notified that the OLE object has moved or resized.
	OnResize (see page 2612)	Occurs immediately after the OLE container window is resized.
	OnStartDrag (see page 2612)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

47.1.1.6.1 TOleContainer.OnActivate Event

Occurs immediately after the OLE object is activated.

Pascal

```
property OnActivate: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnActivate;
```

Description

Write an OnActivate event handler to take specific action when the OLE object becomes active.

See Also

AutoActivate (see page 2557)

DoVerb (see page 2542)

OnDeactivate (see page 2583)

State (see page 2579)

47.1.1.6.2 TOleContainer.OnDeactivate Event

Occurs immediately after the OLE object is deactivated.

Pascal

```
property OnDeactivate: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnDeactivate;
```

Description

Write an OnDeactivate event handler to respond when the OLE object is deactivated.

See Also

Close (see page 2536)

OnActivate (see page 2583)

State (see page 2579)

47.1.1.6.3 TOleContainer.OnDragDrop Event

Occurs when the user drops an object being dragged.

Pascal

```
property OnDragDrop;
```

C++

```
__property OnDragDrop;
```

Description

Use the OnDragDrop event handler to specify what happens when the user drops an object. The Source parameter of the OnDragDrop event is the object being dropped, and the Sender is the control the object is being dropped on. The X and Y parameters are the coordinates of the mouse positioned over the control.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}

void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}
```

```
/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
}
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  TForm1(Source).Left := X;
  TForm1(Source).Top := Y;
end;

procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;
```

```
procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := False;
  if (Source is TLabel) and (State = dsDragEnter) then
    (Source as TLabel).EndDrag(False);
end;
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
dropped label.
}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Not');
  ListBox1.Items.Add('In');
  ListBox1.Items.Add('Alphabetical');
  ListBox1.Items.Add('Order');
end;

// This OnDragOver event handler permits the list box to
// accept a dropped label:

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

// This OnDragDrop event handler implements the drop behavior.

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
    with Sender as TListBox do
    begin
      Font := (Source as TLabel).Font;
      Color := (Source as TLabel).Color;
      DoubleBuffered := true;
      Color := clWindow;
    end;
  end;
end;
```

See Also

BeginDrag

DragDrop

Dragging

EndDrag

OnDragOver

OnEndDrag

OnStartDrag

TDragObject

47.1.1.6.4 TOleContainer.OnDragOver Event

Occurs when the user drags an object over a control.

Pascal

```
property OnDragOver;
```

C++

```
__property OnDragOver;
```

Description

Use an OnDragOver event to signal that the control can accept a dragged object so the user can drop or dock it.

Within the OnDragOver event handler, change the Accept parameter to false to reject the dragged object. Leave Accept as true to allow the user to drop or dock the dragged object on the control.

To change the shape of the cursor, indicating that the control can accept the dragged object, change the value of the DragCursor property for the control before the OnDragOver event occurs.

The Source is the object being dragged, the Sender is the potential drop or dock site, and X and Y are screen coordinates in pixels. The State parameter specifies how the dragged object is moving over the control.

Note: Within the OnDragOver event handler, the Accept parameter defaults to true. However, if an OnDragOver event handler is not supplied, the control rejects the dragged object, as if the Accept parameter were changed to false.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}

void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
```

```
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
}
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
```

```
begin
  TForm1(Source).Left := X;
  TForm1(Source).Top := Y;
end;

procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := False;
  if (Source is TLabel) and (State = dsDragEnter) then
    (Source as TLabel).EndDrag(False);
end;
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
dropped label.
}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Not');
  ListBox1.Items.Add('In');
  ListBox1.Items.Add('Alphabetical');
  ListBox1.Items.Add('Order');
end;

// This OnDragOver event handler permits the list box to
// accept a dropped label:

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

// This OnDragDrop event handler implements the drop behavior.

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
    with Sender as TListBox do
    begin
      Font := (Source as TLabel).Font;
      Color := (Source as TLabel).Color;
      DoubleBuffered := true;
      Color := clWindow;
    end;
  end;
end;
```

See Also

DragCursor

OnDragDrop

OnEndDrag

OnStartDrag

TDragObject

DragKind

OnEndDock

OnDockDrop

OnStartDock

47.1.1.6.5 TOleContainer.OnEndDrag Event

Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.

Pascal

```
property OnEndDrag;
```

C++

```
__property OnEndDrag;
```

Description

Use the OnEndDrag event handler to specify any special processing that occurs when dragging stops.

C++ Examples:

```
/*
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
*/
void __fastcall TForm1::Label1EndDrag(TObject *Sender, TObject *Target, int X,
      int Y)
{
  if (Sender->ClassNameIs("TLabel"))
  {
    AnsiString S = (dynamic_cast<TLabel *>(Sender))->Name + " was dropped and ";
    if (Target)
      S = S + "accepted!";
    else
      S = S + "rejected.";
    Status->Caption = S;
  }
}
```

Delphi Examples:

```
{
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
}
procedure TForm1.Label1EndDrag(Sender, Target: TObject; X, Y: Integer);
var
  S: string;
begin
  S := (Sender as TLabel).Name + ' was dropped... and ';
  if Target <> nil then
    S := S + 'accepted!'
  else
    S := S + 'rejected!';
```

```
  Status.Caption := S;
end;
```

See Also

OnDragDrop

OnDragOver

OnStartDrag

DragKind

47.1.1.6.6 TOleContainer.OnEnter Event

Occurs when a control receives the input focus.

Pascal

```
property OnEnter;
```

C++

```
__property OnEnter;
```

Description

Use the OnEnter event handler to cause any special processing to occur when a control becomes active.

The OnEnter event does not occur when switching between forms or between another application and the application that includes the control.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnEnter event occurs for the container before the OnEnter event of the contained control.

Similarly, an OnExit event of the container occurs after the OnExit event of the control in a container when focus moves to another control outside the container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

C++ Examples:

```
/*
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
*/
TWinControl *myActiveControl;

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  FontDialog1->Options << fdApplyButton;
  FontDialog1->Execute();
}

void __fastcall TForm1::FontDialog1Apply(TObject *Sender, HWND Wnd)
```

```
{
  if (myActiveControl->ClassNameIs("TEdit"))
   (dynamic_cast<TEdit *>(myActiveControl))->Font->Assign(FontDialog1->Font);
  else if (myActiveControl->ClassNameIs("TRichEdit"))
    (dynamic_cast<TRichEdit *>(myActiveControl))->SelAttributes->Assign(FontDialog1->Font);
  else
    MessageBeep(0);
}

void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  myActiveControl = ActiveControl;
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
   FontDialog1.Options := FontDialog1.Options + [fdApplyButton];
   FontDialog1.Execute;
end;

procedure TForm1.Edit1Enter(Sender: TObject);
begin
  myActiveControl := ActiveControl;
end;

procedure TForm1.FontDialog1Apply(Sender: TObject; Wnd: HWND);
begin
  if myActiveControl is TEdit then
```

```
      with myActiveControl as TEdit do
         Font.Assign(TFontDialog(Sender).Font)
    else if myActiveControl is TRichEdit then
      with myActiveControl as TRichEdit do
        SelAttributes.Assign(TFontDialog(Sender).Font)
    else
      Beep;
  end;
  {
  This example uses an edit box and a memo control on a form.
  When either the edit box or the memo is the active control,
  it is colored yellow. When the active control becomes
  inactive, the color of the control returns to the Windows
  system color for a window.  These event handlers could also
  be shared.
  }
  procedure TForm1.Edit1Enter(Sender: TObject);
  begin
    Edit1.Color := clYellow;
  end;

  procedure TForm1.Edit1Exit(Sender: TObject);
  begin
    Edit1.Color := clWindow;
  end;

  procedure TForm1.Memo1Enter(Sender: TObject);
  begin
    Memo1.Color := clYellow;
  end;

  procedure TForm1.Memo1Exit(Sender: TObject);
  begin
    Memo1.Color := clWindow;
  end;
```

See Also

OnExit

ActiveControl

OnActivate

Using a Single Navigator for Multiple Datasets

47.1.1.6.7 TOleContainer.OnExit Event

Occurs when the input focus shifts away from one control to another.

Pascal

```
property OnExit;
```

C++

```
__property OnExit;
```

Description

Use the OnExit event handler to provide special processing when the control ceases to be active.

The OnExit event does not occur when switching between forms or between another application and your application.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnExit event occurs for the control inside the container before the OnExit event of the container.

Similarly, an OnEnter event of the container occurs before the OnEnter event of the control in a container when focus moves to a control inside a container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

Note: In some control classes the ActiveControl property updates before the OnExit event occurs.

C++ Examples:

```
/*
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255 ,
as this validation test cannot be accomplished
using the edit mask property alone.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  // initialize the maskedit boxes to contain the empty string
  MaskEdit1->Text = "";
  MaskEdit2->Text = "";
  MaskEdit3->Text = "";

  // use the Courier font in the maskedit boxes
  MaskEdit1->Font->Name = "Courier";
  MaskEdit2->Font->Name = "Courier";
  MaskEdit3->Font->Name = "Courier";

  // initialize the mask of each maskedit box
  // according to IP address coding conventions
  MaskEdit1->EditMask = "!099.099.099.099;1; ";
  MaskEdit2->EditMask = "!099.099.099.099;1; ";
  MaskEdit3->EditMask = "!099.099.099.099;1; ";

  MaskEdit2->OnExit = MaskEdit2Exit;
}

void __fastcall TForm1::MaskEdit2Exit(TObject *Sender)
{
  // extract the net and host address from the IP
  String IP = MaskEdit1->Text;
  int net1 = IP.SubString(0, 3).TrimRight().ToInt();
  int net2 = IP.SubString(5, 3).TrimRight().ToInt();
  int host1 = IP.SubString(9, 3).TrimRight().ToInt();
  int host2 = IP.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (net1 < 0 || net1 > 255 ||
      net2 < 0 || net2 > 255 ||
      host1 < 0 || host1 > 255 ||
      host2 < 0 || host2 > 255)
```

```
      throw(Exception("Not a valid IP address."));

    // extract the net and host mask from the subnet mask
    String mask = MaskEdit2->Text;
    int netmask1 = mask.SubString(0, 3).TrimRight().ToInt();
    int netmask2 = mask.SubString(5, 3).TrimRight().ToInt();
    int hostmask1 = mask.SubString(9, 3).TrimRight().ToInt();
    int hostmask2 = mask.SubString(13, 3).TrimRight().ToInt();

    // a range test that you cannot validate through edit masks
    if (netmask1 < 0 || netmask1 > 255 ||
        netmask2 < 0 || netmask2 > 255 ||
        hostmask1 < 0 || hostmask1 > 255 ||
        hostmask2 < 0 || hostmask2 > 255)
      throw(Exception("Not a valid subnet mask."));

    // compute the subnet address using bit-wise AND
    int sub_net1 = net1 & netmask1;
    int sub_net2 = net2 & netmask2;
    int sub_host1 = host1 & hostmask1;
    int sub_host2 = host2 & hostmask2;

    // display the subnet address
    MaskEdit3->Text =
      String(sub_net1) + "." + String(sub_net2) + "." +
      String(sub_host1) + "." + String(sub_host2);
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
```

```
  An additional test is made to assure that the values in the IP address
  and in the subnet mask are in the interval from 0 to 255,
  as this validation test cannot be accomplished
  using the edit mask property alone.
  }
  procedure TForm1.FormCreate(Sender: TObject);
  begin
    { initialize the maskedit boxes to contain the empty string }
    MaskEdit1.Text := '';
    MaskEdit2.Text := '';
    MaskEdit3.Text := '';

    { use the Courier font in the maskedit boxes }
    MaskEdit1.Font.Name := 'Courier';
    MaskEdit2.Font.Name := 'Courier';
    MaskEdit3.Font.Name := 'Courier';

    {
    initialize the mask of each maskedit box
    according to IP address coding conventions
    }
    MaskEdit1.EditMask := '!099.099.099.099;1; ';
    MaskEdit2.EditMask := '!099.099.099.099;1; ';
    MaskEdit3.EditMask := '!099.099.099.099;1; ';

    MaskEdit2.OnExit := MaskEdit2Exit;
  end;

  procedure TForm1.MaskEdit2Exit(Sender: TObject);
  var
    net1, net2, host1, host2,
    netmask1, netmask2, hostmask1, hostmask2,
    sub_net1, sub_net2, sub_host1, sub_host2: Integer;

    IP, mask: String;

  begin
    // extract the net and host address from the IP
    IP := MaskEdit1.Text;
    net1 := StrToInt(TrimRight(Copy(IP, 0, 3)));
    net2 := StrToInt(TrimRight(Copy(IP, 5, 3)));
    host1 := StrToInt(TrimRight(Copy(IP, 9, 3)));
    host2 := StrToInt(TrimRight(Copy(IP, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((net1 < 0) Or (net1 > 255) Or
        (net2 < 0) Or (net2 > 255) Or
        (host1 < 0) Or (host1 > 255) Or
        (host2 < 0) Or (host2 > 255)) then
      raise EArgumentException.Create('Not a valid IP address.');

    // extract the net and host mask from the subnet mask
    mask := MaskEdit2.Text;
    netmask1 := StrToInt(TrimRight(Copy(mask, 0, 3)));
    netmask2 := StrToInt(TrimRight(Copy(mask, 5, 3)));
    hostmask1 := StrToInt(TrimRight(Copy(mask, 9, 3)));
    hostmask2 := StrToInt(TrimRight(Copy(mask, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((netmask1 < 0) Or (netmask1 > 255) Or
        (netmask2 < 0) Or (netmask2 > 255) Or
        (hostmask1 < 0) Or (hostmask1 > 255) Or
        (hostmask2 < 0) Or (hostmask2 > 255)) then
      raise EArgumentException.Create('Not a valid subnet mask.');

    // compute the subnet address using bit-wise AND
```

```
    sub_net1 := net1 And netmask1;
    sub_net2 := net2 And netmask2;
    sub_host1 := host1 And hostmask1;
    sub_host2 := host2 And hostmask2;

    // display the subnet address
    MaskEdit3.Text :=
      IntToStr(sub_net1) + '.' + IntToStr(sub_net2) + '.' +
      IntToStr(sub_host1) + '.' + IntToStr(sub_host2);
  end;
  {
  This example uses an edit box and a memo control on a form.
  When either the edit box or the memo is the active control,
  it is colored yellow. When the active control becomes
  inactive, the color of the control returns to the Windows
  system color for a window.  These event handlers could also
  be shared.
  }
  procedure TForm1.Edit1Enter(Sender: TObject);
  begin
    Edit1.Color := clYellow;
  end;

  procedure TForm1.Edit1Exit(Sender: TObject);
  begin
    Edit1.Color := clWindow;
  end;

  procedure TForm1.Memo1Enter(Sender: TObject);
  begin
    Memo1.Color := clYellow;
  end;

  procedure TForm1.Memo1Exit(Sender: TObject);
  begin
    Memo1.Color := clWindow;
  end;
```

See Also

OnEnter

ActiveControl

47.1.1.6.8 TOleContainer.OnKeyDown Event

Occurs when a user presses any key while the control has focus.

Pascal

```
property OnKeyDown;
```

C++

```
__property OnKeyDown;
```

Description

Use the OnKeyDown event handler to specify special processing to occur when a key is pressed. The OnKeyDown handler can respond to keyboard keys, including function keys and keys combined with the `Shift`, `Alt`, and `Ctrl` keys, and pressed mouse buttons.

Note: TCheckBox does not fire OnKeyDown events for arrow keys.

The TKeyEvent type points to a method that handles keyboard events.

The Key parameter is the key on the keyboard. For non-alphanumeric keys, use virtual key codes to determine the key pressed.

For more information, see Virtual Key codes.

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

An application gets Windows WM_KEYDOWN messages for all keys when the user presses a key. These messages indirectly fire the OnKeyDown event. If you set the Key to 0 there you prevent only any further processing of this message. But for keys that generate characters Windows also produces WM_CHAR. At the time your OnKeyDown event fires, the WM_CHAR message for the key will already be in the message queue. Setting Key to 0 does not stop it from being delivered, so it fires the OnKeyPress event. You must set the Key to #0 to prevent the control from intercepting the WM_CHAR message.

This method of organizing key processing has advantages. Code that only deals with characters, including control characters like #13 for carriage return, #3 for Ctrl-C and so on, should go into the OnKeyPress event. Code that deals with keys that do not generate characters should be put into the OnKeyDown event.

C++ Examples:

```
/*
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
*/
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  if (Key == VK_F1)
  {
    FormColor = Form1->Color;
    Form1->Color = clAqua;
    if (ComboBox1->Text == "KeyDown")
      ComboBox1->Text = "KeyDown2";
    else
      ComboBox1->Text = "KeyDown";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}

void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key, TShiftState Shift)
{
  if (Key == VK_F1)
  {
    Form1->Color = FormColor;
    if (ComboBox1->Text == "KeyUp")
      ComboBox1->Text = "KeyUp2";
    else
      ComboBox1->Text = "KeyUp";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}
/*
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
*/
#include <Printers.hpp>

void __fastcall TForm1::FormCreate(TObject *Sender)
{
   KeyPreview = True;
```

```
}

void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)

{
  if (Key == VK_ESCAPE && Printer()->Printing)
  {
    Printer()->Abort();
    MessageDlg("Printing aborted", mtInformation,
               TMsgDlgButtons() << mbOK,0);
  }
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMemo *Memo1 = new TMemo(Form1); // The parent (Form1) will clean up this object.
  Memo1->Parent = Form1;
  Memo1->Visible = True;
  Memo1->Width = 700;
  if (OpenDialog1->Execute())
  {
    Memo1->Lines->LoadFromFile(OpenDialog1->FileName);

    Printer()->BeginDoc();
    int X = 200;
    int Y = 200;
    for (int I = 0; I <= 10; I++)
      if (!Printer()->Aborted)
      {
        Printer()->Canvas->TextOut(X, Y, Memo1->Lines->Strings[I]);
        Y = Y + 80;
        if (Y > (Printer()->PageHeight - 300))
        {
          Y = 200;
          Printer()->NewPage();
          Sleep(1000);  // to give you time to abort!
        }
      }
    if (!Printer()->Aborted) Printer()->EndDoc();
  }
}
```

Delphi Examples:

```
{
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
}
var
  FormColor: TColor;
procedure TForm1.FormCreate(Sender: TObject);
begin
  KeyPreview := True;
  FormColor := Form1.Color;
end;

procedure TForm1.FormKeyDown(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := clAqua;
```

```
    if (ComboBox1.Text = 'KeyDown') then
      ComboBox1.Text := 'KeyDown2'
    else
      ComboBox1.Text := 'KeyDown';
    WriteLn(ComboBox1.Text);
  end;
end;

procedure TForm1.FormKeyUp(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := FormColor;
    if (ComboBox1.Text = 'KeyUp') then
      ComboBox1.Text := 'KeyUp2'
    else
      ComboBox1.Text := 'KeyUp';
    WriteLn(ComboBox1.Text);
  end;
end;
{
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
}

uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  I, X, Y: Integer;
  Memo1 : TMemo;
  r: TRect;
begin
  Memo1 := TMemo.Create(Form1);
  Memo1.Parent := Form1;
  Memo1.Visible := True;
  Memo1.Width := 700;
  if (OpenDialog1.Execute) then
  begin
    Memo1.Lines.LoadFromFile(OpenDialog1.FileName);

    Printer.BeginDoc;
    X := 200;
    Y := 200;
    for I := 0 to 140 do
      if (not Printer.Aborted) then
      begin
        Printer.Canvas.TextOut(X, Y, Memo1.Lines[I]);
        Y := Y + 80;
        if (Y > (Printer.PageHeight - 300)) then
        begin
          Y := 200;
          Printer.NewPage;
          Sleep(1000);  // to give you time to abort!
        end;
      end;
    if (not Printer.Aborted) then Printer.EndDoc;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
   KeyPreview := True;
end;
```

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
if (Key=VK_ESCAPE) and Printer.Printing then
  begin
  Printer.Abort;
  MessageDlg('Printing aborted', mtInformation, [mbOK],0);
  end;
end;
```

See Also

KeyPreview

OnKeyPress

OnKeyUp

47.1.1.6.9 TOleContainer.OnKeyPress Event

Occurs when a key is pressed.

Pascal

```
property OnKeyPress;
```

C++

```
__property OnKeyPress;
```

Description

Use the OnKeyPress event handler to make something happen as a result of a single character key press.

The Key parameter in the OnKeyPress event handler is of type Char; therefore, the OnKeyPress event registers the ASCII character of the key pressed. Keys that don't correspond to an ASCII Char value (Shift or F1, for example) don't generate an OnKeyPress event. Key combinations (such as Shift+A), generate only one OnKeyPress event (for this example, Shift+A results in a Key value of "A" if Caps Lock is off). To respond to non-ASCII keys or key combinations, use the OnKeyDown or OnKeyUp event handlers.

An application gets Windows WM_KEYDOWN messages for all keys when the user presses a key. These messages indirectly fire the OnKeyDown event. If you set the Key to 0 there you prevent only any further processing of this message. But for keys that generate characters Windows also produces WM_CHAR. At the time your OnKeyDown event fires, the WM_CHAR message for the key will already be in the message queue. Setting Key to 0 does not stop it from being delivered, so it fires the OnKeyPress event. You must set the Key to #0 to prevent the control from intercepting the WM_CHAR message.

This method of organizing key processing has advantages. Code that only deals with characters, including control characters like #13 for carriage return, #3 for Ctrl-C and so on, should go into the OnKeyPress event. Code that deals with keys that do not generate characters should be put into the OnKeyDown event.

C++ Examples:

```
/*
This event handler displays a message dialog box specifying
which key was pressed:
*/

void __fastcall TForm1::FormKeyPress(TObject *Sender, char &Key)
{
  char keyString[25];
  keyString[0] = Key;
  strcpy(&keyString[1], " Was Pressed");
  TMsgDlgButtons() << mbOK;
  MessageDlg(AnsiString(keyString), TMsgDlgType(mtInformation), TMsgDlgButtons() << mbOK, 0);
```

```
}
```

Delphi Examples:

```
{
This event handler displays a message dialog box specifying
which key was pressed:
}
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  MessageDlg(Key + ' has been pressed', mtInformation, [mbOK], 0)
end;
```

See Also

OnKeyDown

OnKeyUp

KeyPreview

47.1.1.6.10 TOleContainer.OnKeyUp Event

Occurs when the user releases a key that has been pressed.

Pascal

```
property OnKeyUp;
```

C++

```
__property OnKeyUp;
```

Description

Use the OnKeyUp event handler to provide special processing that occurs when a key is released. The OnKeyUp handler can respond to all keyboard keys, including function keys and keys combined with the Shift, Alt, and Ctrl keys.

The TKeyEvent type points to a method that handles keyboard events. The Key parameter is the key on the keyboard. For non-alphanumeric keys, you must use virtual key codes to determine the key pressed. For more information, see Virtual Key codes.

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

C++ Examples:

```
/*
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
*/
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  if (Key == VK_F1)
  {
    FormColor = Form1->Color;
    Form1->Color = clAqua;
    if (ComboBox1->Text == "KeyDown")
      ComboBox1->Text = "KeyDown2";
    else
      ComboBox1->Text = "KeyDown";
    Form1->Refresh();
```

```
    Memo1->Lines->Add(ComboBox1->Text);
  }
}

void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key, TShiftState Shift)
{
  if (Key == VK_F1)
  {
    Form1->Color = FormColor;
    if (ComboBox1->Text == "KeyUp")
      ComboBox1->Text = "KeyUp2";
    else
      ComboBox1->Text = "KeyUp";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}
```

Delphi Examples:

```
{
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
}
var
  FormColor: TColor;
procedure TForm1.FormCreate(Sender: TObject);
begin
  KeyPreview := True;
  FormColor := Form1.Color;
end;

procedure TForm1.FormKeyDown(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := clAqua;
    if (ComboBox1.Text = 'KeyDown') then
      ComboBox1.Text := 'KeyDown2'
    else
      ComboBox1.Text := 'KeyDown';
    WriteLn(ComboBox1.Text);
  end;
end;

procedure TForm1.FormKeyUp(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := FormColor;
    if (ComboBox1.Text = 'KeyUp') then
      ComboBox1.Text := 'KeyUp2'
    else
      ComboBox1.Text := 'KeyUp';
    WriteLn(ComboBox1.Text);
  end;
end;
```

See Also

OnKeyDown

OnKeyPress

47.1.1.6.11 TOleContainer.OnMouseActivate Event

Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.

Pascal

```
property OnMouseActivate;
```

C++

```
__property OnMouseActivate;
```

Description

Use the OnMouseActivate event handler to implement any special processing that should occur as a result of pressing a mouse button on a control when the parent top-level form is not active.

The event type is:

```
TMouseActivateEvent = procedure (Sender: TObject; Button: TMouseButton; ShiftState:
TShiftState;
X, Y: Integer; HitTest: Integer; var MouseActivate: TMouseActivate) of object;
```

When you click on a control and the parent top-level form is not active, this event fires on the control that the mouse cursor is over. MouseActivate is a protected virtual function in TControl. MouseActivate can be overridden in custom control descendants to provide special built-in processing of the OnMouseActivate events. If you leave the default value of MouseActivate as maDefault, the parent control fires the OnMouseActivate event and repeats the process all the way up to the top-level form. If no control in the chain sets the value of MouseActivate, the behavior is the same as if a control set TMouseActivate to maActivate. At any point, a control can set the value of TMouseActivate and the parent processing stops.

Some controls set focus to themselves when they get a button down message so, in some cases, setting maNoActivate appears to have no effect. For instance, a TButton control sets focus to itself in the WM_LBUTTONDOWN message regardless of whether or not TMouseActivate is set to maNoActivate. In this case, setting maNoActivateAndEat will work because the top-level form is not activated and the WM_LBUTTONDOWN message is suppressed.

The HitTest parameter is the hit test value obtained from the WM_NCHITTEST message. See Windows.pas for valid values and their meanings. In most cases this value will be HTCLIENT, which means that the user clicked in the client area of a control. However, in the case of a top-level form, this can take other values such as HTCAPTION, or HTBOTTOM. This allows the code to decide which value to set for MouseActivate, based on the HitTest code. For example, the following code in the OnMouseActivate event handler for a TForm forces the user to click into a non-client area to activate the form:

```
if HitTest = HTCLIENT then
    MouseActivate := maNoActivateAndEat;
```

Even though this event is tied to how Windows processes WM_MOUSEACTIVATE, the actual handling of the message is done at the TControl level, which is the point in the VCL hierarchy where the control has no window handle. The OnMouseActivate event is then simulated since the all TControl descendants also have OnMouseDown, OnMouseMove, and OnMouseUp events. This information is applicable to both VCL Win32 and VCL.NET.

See Also

OnClick

OnMouseMove

OnMouseUp

MouseCapture

47.1.1.6.12 TOleContainer.OnMouseDown Event

Occurs when the user presses a mouse button with the mouse pointer over a control.

Pascal

```
property OnMouseDown;
```

C++

```
__property OnMouseDown;
```

Description

Use the OnMouseDown event handler to implement any special processing that should occur as a result of pressing a mouse button.

The OnMouseDown event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. X and Y are the pixel coordinates of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
```

```
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  Form1.Canvas.Rectangle(StartX, StartY, X, Y);
  StatusBar1.Panels[0].Text := '';
  StatusBar1.Panels[1].Text := '';
  StatusBar1.Panels[2].Text := '';
  StatusBar1.Panels[3].Text := '';
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if ssLeft in Shift then { make sure button is down }
  begin
    if Y > StartY then
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
      end
    else
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
      end;
    if X > StartX then
```

```
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
        end
      else
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
        end;
    end;
  end;
```

See Also

OnClick

OnMouseMove

OnMouseUp

MouseCapture

47.1.1.6.13 TOleContainer.OnMouseMove Event

Occurs when the user moves the mouse pointer while the mouse pointer is over a control.

Pascal

```
property OnMouseMove;
```

C++

```
__property OnMouseMove;
```

Description

Use the OnMouseMove event handler to respond when the mouse pointer moves after the control has captured the mouse.

Use the Shift parameter of the OnMouseDown event handler, to determine to the state of the shift keys and mouse buttons. Shift keys are the Shift, Ctrl, and Alt keys or shift key-mouse button combinations. X and Y are pixel coordinates of the new location of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
```

```
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  Form1.Canvas.Rectangle(StartX, StartY, X, Y);
  StatusBar1.Panels[0].Text := '';
  StatusBar1.Panels[1].Text := '';
  StatusBar1.Panels[2].Text := '';
```

```
    StatusBar1.Panels[3].Text := '';
  end;

  procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
  begin
    if ssLeft in Shift then { make sure button is down }
    begin
      if Y > StartY then
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
        end
      else
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
        end;
      if X > StartX then
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
        end
      else
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
        end;
    end;
  end;
```

See Also

OnMouseDown

OnMouseUp

MouseCapture

47.1.1.6.14 TOleContainer.OnMouseUp Event

Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.

Pascal

```
property OnMouseUp;
```

C++

```
__property OnMouseUp;
```

Description

Use an OnMouseUp event handler to implement special processing when the user releases a mouse button.

The OnMouseUp event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. X and Y are the pixel coordinates of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
```

```
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
```

```
  the rectangle are displayed in the status bar.
  }
  procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
  Y: Integer);
  begin
    StartX := X;
    StartY := Y;
  end;

  procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
  Integer);
  begin
    Form1.Canvas.Rectangle(StartX, StartY, X, Y);
    StatusBar1.Panels[0].Text := '';
    StatusBar1.Panels[1].Text := '';
    StatusBar1.Panels[2].Text := '';
    StatusBar1.Panels[3].Text := '';
  end;

  procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
  begin
    if ssLeft in Shift then { make sure button is down }
    begin
      if Y > StartY then
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
        end
      else
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
        end;
      if X > StartX then
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
        end
      else
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
        end;
    end;
  end;
```

See Also

OnMouseDown

OnMouseMove

Responding to a Mouse-up Action

47.1.1.6.15 TOleContainer.OnObjectMove Event

Occurs when the OLE container is notified that the OLE object has moved or resized.

Pascal

```
property OnObjectMove: TObjectMoveEvent;
```

C++

```
__property TObjectMoveEvent OnObjectMove;
```

Description

Write an OnObjectMove event handler to respond when the OLE object changes size or position. The Bounds parameter specifies the boundaries of the in-place object in the client coordinates of OLE container. OnObjectMove occurs before the OLE object's image is updated to reflect the change.

See Also

OnResize (see page 2612)

47.1.1.6.16 TOleContainer.OnResize Event

Occurs immediately after the OLE container window is resized.

Pascal

```
property OnResize: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnResize;
```

Description

Write an OnResize event handler to respond to changes in the size of the OLE container window. If SizeMode (see page 2578) is smAutoSize, the container window changes size whenever the OLE object resizes.

See Also

OnObjectMove (see page 2611)

SizeMode (see page 2578)

47.1.1.6.17 TOleContainer.OnStartDrag Event

Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

Pascal

```
property OnStartDrag;
```

C++

```
__property OnStartDrag;
```

Description

Use the OnStartDrag event handler to implement special processing when the user starts to drag the control or an object it contains. OnStartDrag only occurs if DragKind is dkDrag.

Sender is the control that is about to be dragged, or that contains the object about to be dragged.

The OnStartDrag event handler can create a TDragControlObjectEx instance for the DragObject parameter to specify the drag cursor, or, optionally, a drag image list. If you create a TDragControlObjectEx instance, there is no need to call the Free method for the DragObject when dragging is over. If you create, instead, a TDragControlObject instance, your application is responsible for freeing the drag object instance.

If the OnStartDrag event handler sets the DragObject parameter to nil (Delphi) or NULL (C++), a TDragControlObject object is automatically created and dragging begins on the control itself.

Note: On some controls, such as TRichEdit, the underlying Windows control handles internal drag operations. For these controls, there are no OnStartDrag or OnEndDrag events for drag operations within the control.

See Also

OnEndDrag

TDragControlObjectEx

DragKind

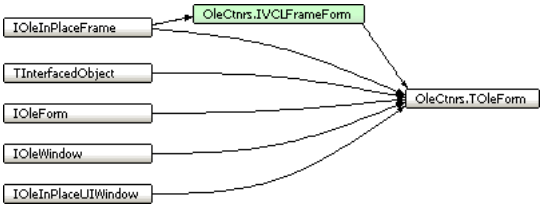
OnDragOver

OnDragDrop

47.1.2 TOleForm Class

TOleForm is an object that provides OLE support for a TForm object.

Class Hierarchy



Pascal

```
TOleForm = class(TInterfacedObject, IOleForm, IOleWindow, IOleInPlaceUIWindow,
IOleInPlaceFrame, IVCLFrameForm);
```

C++

```
class TOleForm : public TInterfacedObject, public IOleForm, public IOleWindow, public
IOleInPlaceUIWindow, public IOleInPlaceFrame, public IVCLFrameForm;
```

File

OleCtnrs

Description

TOleForm implements the IOleForm, IOleWindow, IOleInPlaceUIWindow, and IOleInPlaceFrame interfaces, as well as an interface for internal use. The implementation of the TOleForm object is used internally by the Visual Component Library.

47.1.2.1 TOleForm Members

The following tables list the members exposed by TOleForm.

Public Methods

	Name	Description
	AddContainer (see page 2616)	This is AddContainer, a member of class IVCLFrameForm.
	ClearBorderSpace (see page 2616)	This is ClearBorderSpace, a member of class IVCLFrameForm.
	Create (see page 2614)	Create creates a new instance of the TOleForm (see page 2613) object.
	Destroy (see page 2614)	Destroy frees the memory associated with this instance of the TOleForm (see page 2613) object.
	Form (see page 2616)	This is Form, a member of class IVCLFrameForm.
	RemoveContainer (see page 2616)	This is RemoveContainer, a member of class IVCLFrameForm.

47.1.2.2 TOleForm.Create Constructor

Create creates a new instance of the TOleForm (see page 2613) object.

Pascal

```
constructor Create(Form: TCustomForm);
```

C++

```
__fastcall TOleForm(TCustomForm Form);
```

Description

Create is used to create a new instance of the TOleForm (see page 2613) object. Form (see page 2616) is the form upon which to base the new TOleForm (see page 2613) object.

See Also

Destroy (see page 2614)

47.1.2.3 TOleForm.Destroy Destructor

Destroy frees the memory associated with this instance of the TOleForm (see page 2613) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TOleForm();
```

Description

Do not call Destroy directly. Instead, call Free, which calls Destroy if this instance of the TOleForm (see page 2613) object is not nil.

Destroy frees private objects allocated and used by this instance of the TOleForm (see page 2613) object.

See Also

Create (see page 2614)

47.2 Interfaces

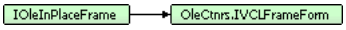
The following table lists interfaces in this documentation.

Interfaces

	Name	Description
	IVCLFrameForm (see page 2615)	This is class OleCtnrs.IVCLFrameForm.

47.2.1 IVCLFrameForm Interface

Class Hierarchy



Pascal

```
IVCLFrameForm = interface(IOleInPlaceFrame);
```

C++

```
__interface IVCLFrameForm : public IOleInPlaceFrame;
```

File

OleCtnrs

Description

This is class OleCtnrs.IVCLFrameForm.

47.2.1.1 IVCLFrameForm Members

The following tables list the members exposed by IVCLFrameForm.

Public Methods

	Name	Description
	AddContainer (see page 2616)	This is AddContainer, a member of class IVCLFrameForm.
	ClearBorderSpace (see page 2616)	This is ClearBorderSpace, a member of class IVCLFrameForm.
	Form (see page 2616)	This is Form, a member of class IVCLFrameForm.
	RemoveContainer (see page 2616)	This is RemoveContainer, a member of class IVCLFrameForm.

47.2.1.2 IVCLFrameForm Methods

The methods of the IVCLFrameForm class are listed here.

Public Methods

	Name	Description
	AddContainer (see page 2616)	This is AddContainer, a member of class IVCLFrameForm.
	ClearBorderSpace (see page 2616)	This is ClearBorderSpace, a member of class IVCLFrameForm.
	Form (see page 2616)	This is Form, a member of class IVCLFrameForm.

	RemoveContainer (see page 2616)	This is RemoveContainer, a member of class IVCLFrameForm.

47.2.1.2.1 IVCLFrameForm.AddContainer Method

Pascal

```
procedure AddContainer(Instance: TOleContainer);
```

C++

```
__fastcall AddContainer(TOleContainer Instance);
```

Description

This is AddContainer, a member of class IVCLFrameForm.

47.2.1.2.2 IVCLFrameForm.ClearBorderSpace Method

Pascal

```
procedure ClearBorderSpace;
```

C++

```
__fastcall ClearBorderSpace();
```

Description

This is ClearBorderSpace, a member of class IVCLFrameForm.

47.2.1.2.3 IVCLFrameForm.Form Method

Pascal

```
function Form: TCustomForm;
```

C++

```
__fastcall TCustomForm Form();
```

Description

This is Form, a member of class IVCLFrameForm.

47.2.1.2.4 IVCLFrameForm.RemoveContainer Method

Pascal

```
procedure RemoveContainer(Instance: TOleContainer);
```

C++

```
__fastcall RemoveContainer(TOleContainer Instance);
```

Description

This is RemoveContainer, a member of class IVCLFrameForm.

47.3 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	DestroyMetaPict (see page 2617)	Frees a metafile resource given its handle.

47.3.1 OleCtnrs.DestroyMetaPict Function

Frees a metafile resource given its handle.

Pascal

```
procedure DestroyMetaPict(MetaPict: HGlobal);
```

C++

```
DestroyMetaPict(HGlobal MetaPict);
```

File

OleCtnrs

Description

Call DestroyMetaPict to free a metafile obtained using the GetIconMetaPict method of TOleContainer (see page 2529).

You do not need to use DestroyMetaPict to free metafiles represented using TMetafile. These are freed automatically when the TMetafile instance is destroyed. However, if you assign a new value to the Handle property of TMetafile, thereby replacing its old metafile, you must free the old metafile by calling DestroyMetaPict.

See Also

TOleContainer (see page 2529)

TMetafile

47.4 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TAutoActivate (see page 2618)	This is record OleCtnrs.TAutoActivate.
	TCreateType (see page 2619)	TCreateType is an enumerated type that defines values for TCreateInfo (see page 2618)'s CreateType field.
	TObjectState (see page 2620)	This is record OleCtnrs.TObjectState.
	TSizeMode (see page 2621)	This is record OleCtnrs.TSizeMode.

Records

	Name	Description
	TCreateInfo (see page 2618)	TCreateInfo is used by TOleContainer (see page 2529)'s CreateObjectFromInfo method.
	TVerbInfo (see page 2621)	This is record OleCtnrs.TVerbInfo.

47.4.1 OleCtnrs.TAutoActivate Enumeration

Pascal

```
TAutoActivate = (
  aaManual,
  aaGetFocus,
  aaDoubleClick
);
```

C++

```
enum TAutoActivate {
  aaManual,
  aaGetFocus,
  aaDoubleClick
};
```

File

OleCtnrs

Description

This is record OleCtnrs.TAutoActivate.

47.4.2 OleCtnrs.TCreateInfo Record

TCreateInfo is used by TOleContainer (see page 2529)'s CreateObjectFromInfo method.

Pascal

```
TCreateInfo = record
  CreateType: TCreateType;
  ShowAsIcon: Boolean;
```

```
    IconMetaPict: HGlobal;
    ClassID: TCLSID;
    FileName: WideString;
    DataObject: IDataObject;
  end;
```

C++

```
  struct TCreateInfo {
    TCreateType CreateType;
    Boolean ShowAsIcon;
    HGlobal IconMetaPict;
    TCLSID ClassID;
    BSTR FileName;
    IDataObject DataObject;
  };
```

File

OleCtnrs

Description

TCreateInfo represents how TOleContainer (see page 2529)'s CreateObjectFromInfo method should instantiate an OLE object. TCreateInfo's fields are described in the following table:

Fields	Meaning
CreateType	Describes the source of the object. See TCreateType (see page 2619) for details.
ShowAsIcon	Indicates whether the object is shown as an icon (true) or displayed as it would be in the server application (false).
IconMetaPict	Specifies a global memory handle to a metafile picture (TMetaFilePict) to display in place of the OLE object's default icon. IconMetaPict is optional; When set to 0, the object's default icon is used (when ShowAsIcon is true).
ClassID	The class identifier of the OLE object to create. ClassID is only used when CreateType is ctNewObject. You can use the ProgIDToClassID function to get the class identifier for an OLE object's class name (or programmatic identifier).
FileName	The name of a file to link or embed as an OLE object. FileName is only used when CreateType is ctFromFile or ctLinkToFile.
DataObject	An OLE data transfer object. DataObject is only used when CreateType is ctFromData or ctLinkFromData.

See Also

TOleContainer (see page 2529)

47.4.3 OleCtnrs.TCreateType Enumeration

TCreateType is an enumerated type that defines values for TCreateInfo (see page 2618)'s CreateType field.

Pascal

```
  TCreateType = (
    ctNewObject,
    ctFromFile,
    ctLinkToFile,
    ctFromData,
    ctLinkFromData
  );
```

C++

```
enum TCreateType {
  ctNewObject,
  ctFromFile,
  ctLinkToFile,
  ctFromData,
  ctLinkFromData
};
```

File

OleCtnrs

Description

TCreateType defines the possible values for TCreateInfo (see page 2618)'s CreateType field, which is passed to TOleContainer (see page 2529)'s CreateObjectFromInfo method. It is defined as follows:

Values	Meaning
ctNewObject	Creates an embedded OLE object given its class identifier (TCreateInfo.ClassID)
ctFromFile	Creates an embedded OLE object from the contents of the given file (TCreateInfo.FileName).
ctLinkToFile	Creates a linked OLE object from the contents of the given file (TCreateInfo.FileName)
ctFromData	Creates an embedded OLE object from the given data transfer object (TCreateInfo.DataObject)
ctLinkFromData	Creates a linked OLE object from the given data transfer object (TCreateInfo.DataObject)

See Also

TOleContainer (see page 2529)

TCreateInfo (see page 2618)

47.4.4 OleCtnrs.TObjectState Enumeration

Pascal

```
TObjectState = (
  osEmpty,
  osLoaded,
  osRunning,
  osOpen,
  osInPlaceActive,
  osUIActive
);
```

C++

```
enum TObjectState {
  osEmpty,
  osLoaded,
  osRunning,
  osOpen,
  osInPlaceActive,
  osUIActive
};
```

File

OleCtnrs

Description

This is record OleCtnrs.TObjectState.

47.4.5 OleCtnrs.TSizeMode Enumeration

Pascal

```
TSizeMode = (
  smClip,
  smCenter,
  smScale,
  smStretch,
  smAutoSize
);
```

C++

```
enum TSizeMode {
  smClip,
  smCenter,
  smScale,
  smStretch,
  smAutoSize
};
```

File

OleCtnrs

Description

This is record OleCtnrs.TSizeMode.

47.4.6 OleCtnrs.TVerbInfo Record

Pascal

```
TVerbInfo = packed record
  Verb: Smallint;
  Flags: Word;
end;
```

C++

```
struct TVerbInfo {
  Smallint Verb;
  Word Flags;
};
```

File

OleCtnrs

Description

This is record OleCtnrs.TVerbInfo.

47.5 Types

The following table lists types in this documentation.

Types

Name	Description
TObjectMoveEvent (see page 2622)	This is type OleCtnrs.TObjectMoveEvent.

47.5.1 OleCtnrs.TObjectMoveEvent Type

Pascal

```
TObjectMoveEvent = procedure (OleContainer: TOleContainer; const Bounds: TRect) of object;
```

C++

```
(OleContainer: TOleContainer; const Bounds: TRect) ( TObjectMoveEvent)();
```

File

OleCtnrs

Description

This is type OleCtnrs.TObjectMoveEvent.

47.6 Constants

The following table lists constants in this documentation.

Constants

Name	Description
ovDiscardUndoState (see page 2623)	Specifies to discard the undo state.
ovHide (see page 2624)	Specifies to hide an OLE object.
ovInPlaceActivate (see page 2624)	Specifies the activation of an OLE object.
ovOpen (see page 2625)	Specifies the Open action of an OLE object.
ovPrimary (see page 2625)	Specifies the default action of an OLE object.
ovShow (see page 2626)	Specifies to show an OLE object.
ovUIActivate (see page 2627)	Specifies the activation of the UI(User Interface) of an OLE object.

47.6.1 OleCtnrs.ovDiscardUndoState Constant

Specifies to discard the undo state.

Pascal

```
ovDiscardUndoState = -6;
```

C++

```
ovDiscardUndoState = -6;
```

File

OleCtnrs

Description

ovDiscardUndoState is used as a default item of ObjectVerbs.

ovDiscardUndoState is principally used by the DoVerb method in order to discard the undo state of the OLE object.

Note: If the OLE provider cannot process this request, an EOleSysError (Not Implemented) is thrown.

See Also

ovHide (see page 2624)

ovInPlaceActivate (see page 2624)

ovOpen (see page 2625)

ovPrimary (see page 2625)

ovShow (see page 2626)

ovUIActivate (see page 2627)

47.6.2 OleCtnrs.ovHide Constant

Specifies to hide an OLE object.

Pascal

```
ovHide = -3;
```

C++

```
ovHide = -3;
```

File

OleCtnrs

Description

ovHide is used as a default item of ObjectVerbs.

ovHide is principally used by the DoVerb method in order to hide the OLE object.

Note: If the OLE provider cannot process this request, an EOleSysError (Not Implemented) is thrown.

See Also

ovShow (see page 2626)

ovInPlaceActivate (see page 2624)

ovOpen (see page 2625)

ovPrimary (see page 2625)

ovDiscardUndoState (see page 2623)

ovUIActivate (see page 2627)

47.6.3 OleCtnrs.ovInPlaceActivate Constant

Specifies the activation of an OLE object.

Pascal

```
ovInPlaceActivate = -5;
```

C++

```
ovInPlaceActivate = -5;
```

File

OleCtnrs

Description

ovInPlaceActivate is used as a default item of ObjectVerbs.

ovInPlaceActivate is principally used by the DoVerb method in order to activate the OLE object in place.

Note: If the OLE provider cannot process this request, an EOleSysError (Not Implemented) is thrown.

See Also

ovShow (see page 2626)

ovHide (see page 2624)

ovOpen (see page 2625)

ovPrimary (see page 2625)

ovDiscardUndoState (see page 2623)

ovUIActivate (see page 2627)

47.6.4 OleCtnrs.ovOpen Constant

Specifies the Open action of an OLE object.

Pascal

```
ovOpen = -2;
```

C++

```
ovOpen = -2;
```

File

OleCtnrs

Description

ovOpen is used as a default item of ObjectVerbs.

ovOpen is principally used by the DoVerb method in order to open the application window that the OLE provider registered.

Note: If the OLE provider cannot process this request, an EOleSysError (Not Implemented) is thrown.

See Also

ovShow (see page 2626)

ovHide (see page 2624)

ovInPlaceActivate (see page 2624)

ovPrimary (see page 2625)

ovDiscardUndoState (see page 2623)

ovUIActivate (see page 2627)

47.6.5 OleCtnrs.ovPrimary Constant

Specifies the default action of an OLE object.

Pascal

```
ovPrimary = -65536;
```

C++

```
ovPrimary = -65536;
```

File

OleCtnrs

Description

ovPrimary is used as a default item of ObjectVerbs.

ovPrimary is principally used by the DoVerb method in order to send a request for the default action, usually to activate the OLE object.

Note: If the OLE provider cannot process this request, an EOleSysError (Not Implemented) is thrown.

See Also

ovShow (see page 2626)

ovHide (see page 2624)

ovInPlaceActivate (see page 2624)

ovOpen (see page 2625)

ovDiscardUndoState (see page 2623)

ovUIActivate (see page 2627)

47.6.6 OleCtnrs.ovShow Constant

Specifies to show an OLE object.

Pascal

```
ovShow = -1;
```

C++

```
ovShow = -1;
```

File

OleCtnrs

Description

ovShow is used as a default item of ObjectVerbs.

ovShow is principally used by the DoVerb method in order to show the OLE object.

Note: If the OLE provider cannot process this request, an EOleSysError (Not Implemented) is thrown.

See Also

ovHide (see page 2624)

ovInPlaceActivate (see page 2624)

ovOpen (see page 2625)

ovPrimary (see page 2625)

ovDiscardUndoState (see page 2623)

ovUIActivate (see page 2627)

47.6.7 OleCtnrs.ovUIActivate Constant

Specifies the activation of the UI(User Interface) of an OLE object.

Pascal

```
ovUIActivate = -4;
```

C++

```
ovUIActivate = -4;
```

File

OleCtnrs

Description

ovUIActivate is used as a default item of ObjectVerbs.

ovUIActivate is principally used by the DoVerb method in order to activate the User Interface.

Note: If the OLE provider cannot process this request, an EOleSysError (Not Implemented) is thrown.

See Also

ovShow (see page 2626)

ovHide (see page 2624)

ovInPlaceActivate (see page 2624)

ovOpen (see page 2625)

ovDiscardUndoState (see page 2623)

ovPrimary (see page 2625)

48 OleCtrls Namespace

This is namespace OleCtrls.

Topics

Name	Description
Classes (see page 2631)	The following table lists classes in this documentation.
Functions (see page 2699)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 2701)	The following table lists structs, records, enums in this documentation.
Types (see page 2704)	The following table lists types in this documentation.

Classes

	Name	Description
	EOleCtrlError (see page 2631)	EOleCtrlError is the exception class for ActiveX controls.
	TEnumPropDesc (see page 2631)	TEnumPropDesc provides the mapping between strings used to name the values of a property in an OLE control object and the values themselves.
	TEventDispatch (see page 2634)	This is class OleCtrls.TEventDispatch.
	TOleControl (see page 2638)	TOleControl is the base class from which ActiveX controls are derived.

Functions

	Name	Description
	EmptyParam (see page 2699)	This is function OleCtrls.EmptyParam.
	FontToOleFont (see page 2699)	Returns a Variant containing an IFontDispatch interface that represents a TFont object.
	OleFontToFont (see page 2700)	Fills in a TFont structure to represent an IFontDispatch.

Structs, Records, Enums

	Name	Description
	TControlData (see page 2701)	This is record OleCtrls.TControlData.
	TControlData2 (see page 2702)	This is record OleCtrls.TControlData2.
	TEnumValue (see page 2702)	This is record OleCtrls.TEnumValue.

Types

Name	Description
PControlData (see page 2704)	This is type OleCtrls.PControlData.
PControlData2 (see page 2704)	This is type OleCtrls.PControlData2.
PEnumValueList (see page 2705)	This is type OleCtrls.PEnumValueList.
TEnumValueList (see page 2705)	This is type OleCtrls.TEnumValueList.

TOleEnum (see page 2705)	Represents enumerated types that must be sent over a COM interface.
TServiceQuery (see page 2706)	This is type OleCtrls.TServiceQuery.

48.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	EOleCtrlError (see page 2631)	EOleCtrlError is the exception class for ActiveX controls.
	TEnumPropDesc (see page 2631)	TEnumPropDesc provides the mapping between strings used to name the values of a property in an OLE control object and the values themselves.
	TEventDispatch (see page 2634)	This is class OleCtrls.TEventDispatch.
	TOleControl (see page 2638)	TOleControl is the base class from which ActiveX controls are derived.

48.1.1 EOleCtrlError Class

EOleCtrlError is the exception class for ActiveX controls.

Class Hierarchy

Pascal

```
EOleCtrlError = class(Exception);
```

C++

```
class EOleCtrlError : public Exception;
```

File

OleCtrls

Description

EOleCtrlError is raised when an application cannot link to an ActiveX control.

See Also

EOleError

48.1.2 TEnumPropDesc Class

TEnumPropDesc provides the mapping between strings used to name the values of a property in an OLE control object and the values themselves.

Class Hierarchy

Pascal

```
TEnumPropDesc = class;
```

C++

```
class TEnumPropDesc;
```

File

OleCtrls

Description

Use TEnumPropDesc to interpret the strings used by an OLE control object to represent the values of one of its properties. TEnumPropDesc can provide a list of possible value strings to a callback function, or map between the string names of property values and the values themselves. TEnumPropDesc objects are made available through the GetEnumPropDesc method of an OLE control object.

See Also

TOleControl (see page 2638)

48.1.2.1 TEnumPropDesc Members

The following tables list the members exposed by TEnumPropDesc.

Public Methods

	Name	Description
	Create (see page 2632)	Creates an instance of TEnumPropDesc (see page 2631).
	Destroy (see page 2633)	Destroys an instance of TEnumPropDesc (see page 2631).
	GetStrings (see page 2633)	Calls the callback Proc for each property value string.
	StringToValue (see page 2634)	Maps a string representation of the property value to the integer value.
	ValueToString (see page 2634)	Maps an integer that indicates the value of the property to the string representation of that value used by the OLE control object.

48.1.2.2 TEnumPropDesc.Create Constructor

Creates an instance of TEnumPropDesc (see page 2631).

Pascal

```
constructor Create(DispID: Integer; ValueCount: Integer; const TypeInfo: ITypeInfo);
```

C++

```
__fastcall TEnumPropDesc(int DispID, int ValueCount, const ITypeInfo TypeInfo);
```

Description

Create is called by an OLE control object to create an instance of TEnumPropDesc (see page 2631) to represent the values of one of its properties. Applications should not need to create instances of TEnumPropDesc (see page 2631). When working with OLE properties, get the TEnumPropDesc (see page 2631) objects from the OLE control object.

DispID identifies the property of the OLE object, ValueCount is the number of distinct values the property can take TypeInfo is an ITypeInfo object that the constructor can use to capture the values and their strings.

See Also

GetEnumPropDesc (see page 2656)

48.1.2.3 TEnumPropDesc.Destroy Destructor

Destroys an instance of TEnumPropDesc (see page 2631).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TEnumPropDesc();
```

Description

Do not call Destroy directly in an application. The TOleControl (see page 2638) object that owns the TEnumPropDesc (see page 2631) object will destroy it automatically.

Destroy frees the internal list used to map value strings to property values.

See Also

TOleControl (see page 2638)

System::TObject::Free

48.1.2.4 TEnumPropDesc Methods

The methods of the TEnumPropDesc class are listed here.

Public Methods

	Name	Description
	GetStrings (see page 2633)	Calls the callback Proc for each property value string.
	StringToValue (see page 2634)	Maps a string representation of the property value to the integer value.
	ValueToString (see page 2634)	Maps an integer that indicates the value of the property to the string representation of that value used by the OLE control object.

48.1.2.4.1 TEnumPropDesc.GetStrings Method

Calls the callback Proc for each property value string.

Pascal

```
procedure GetStrings(Proc: TGetStrProc);
```

C++

```
__fastcall GetStrings(TGetStrProc Proc);
```

Description

Use GetStrings to execute code for every property value string represented in the TEnumPropDesc (see page 2631) object. The callback Proc is called for every value, with the S parameter set to a string that gives the integer value followed by the string representation of that value.

For example, consider a TEnumPropDesc (see page 2631) object that represents a color property, with values equal to TColor values. The callback Proc would be called for every built-in color name with strings of the form '255 - clRed'.

See Also

StringToValue (see page 2634)

ValueToString (see page 2634)

48.1.2.4.2 TEnumPropDesc.StringToValue Method

Maps a string representation of the property value to the integer value.

Pascal

```
function StringToValue(const S: string): Integer;
```

C++

```
__fastcall int StringToValue(const UnicodeString S);
```

Description

Use StringToValue to map the string representation of a property value used by an OLE control object to an integer value that is a more accurate representation of the property value. If the property value is not an integer, the integer associated with the string can be a pointer to the value.

See Also

ValueToString (see page 2634)

48.1.2.4.3 TEnumPropDesc.ValueToString Method

Maps an integer that indicates the value of the property to the string representation of that value used by the OLE control object.

Pascal

```
function ValueToString(V: Integer): string;
```

C++

```
__fastcall UnicodeString ValueToString(int V);
```

Description

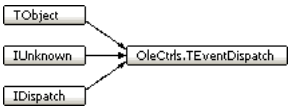
Use ValueToString to map the integer value of a property value to the string representation used by an OLE control object. If the property value is not an integer, the integer associated with the string can be a pointer to the value.

See Also

StringToValue (see page 2634)

48.1.3 TEventDispatch Class

Class Hierarchy



Pascal

```
TEventDispatch = class(TObject, IUnknown, IDispatch);
```

C++

```
class TEventDispatch : public TObject, public IUnknown, public IDispatch;
```

File

OleCtrls

Description

This is class OleCtrls.TEventDispatch.

48.1.3.1 TEventDispatch Members

The following tables list the members exposed by TEventDispatch.

Protected Methods

	Name	Description
	_AddRef (see page 2636)	This is _AddRef, a member of class TEventDispatch.
	_Release (see page 2636)	This is _Release, a member of class TEventDispatch.
	GetIDsOfNames (see page 2636)	GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name.
	GetTypeInfo (see page 2636)	GetTypeInfo returns the object's type information.
	GetTypeInfoCount (see page 2637)	GetTypeInfoCount returns the count of type information interfaces.
V	Invoke (see page 2637)	Invoke performs a call to a late-binded property or method.
	QueryInterface (see page 2638)	This is QueryInterface, a member of class TEventDispatch.

Protected Properties

	Name	Description
R	Control (see page 2638)	This is Control, a member of class TEventDispatch.

Public Methods

	Name	Description
	Create (see page 2635)	This is Create, a member of class TEventDispatch.

48.1.3.2 TEventDispatch.Create Constructor

Pascal

```
constructor Create(Control: TOleControl);
```

C++

```
__fastcall TEventDispatch(TOleControl Control);
```

Description

This is Create, a member of class TEventDispatch.

48.1.3.3 TEventDispatch Methods

The methods of the TEventDispatch class are listed here.

Protected Methods

	Name	Description
	_AddRef (see page 2636)	This is _AddRef, a member of class TEventDispatch.
	_Release (see page 2636)	This is _Release, a member of class TEventDispatch.
	GetIDsOfNames (see page 2636)	GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name.
	GetTypeInfo (see page 2636)	GetTypeInfo returns the object's type information.
	GetTypeInfoCount (see page 2637)	GetTypeInfoCount returns the count of type information interfaces.
V	Invoke (see page 2637)	Invoke performs a call to a late-binded property or method.

	QueryInterface (see page 2638)	This is QueryInterface, a member of class TEventDispatch.

48.1.3.3.1 TEventDispatch._AddRef Method

Pascal

```
function _AddRef: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _AddRef();
```

Description

This is _AddRef, a member of class TEventDispatch.

48.1.3.3.2 TEventDispatch._Release Method

Pascal

```
function _Release: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _Release();
```

Description

This is _Release, a member of class TEventDispatch.

48.1.3.3.3 TEventDispatch.GetIDsOfNames Method

GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name.

Pascal

```
function GetIDsOfNames(const IID: TGUID; Names: Pointer; NameCount: Integer; LocaleID:
Integer; DispIDs: Pointer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetIDsOfNames(const TGUID IID, void * Names, int NameCount, int
LocaleID, void * DispIDs);
```

Description

GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name. GetIDsOfNames is used primarily to obtain dispatch identifiers for the late-binded methods exposed by the COM object through the IDispatch interface.

See Also

GetTypeInfoCount

GetTypeInfo

Invoke

48.1.3.3.4 TEventDispatch.GetTypeInfo Method

GetTypeInfo returns the object's type information.

Pascal

```
function GetTypeInfo(Index: Integer; LocaleID: Integer; out TypeInfo): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetTypeInfo(int Index, int LocaleID,  TypeInfo);
```

Description

GetTypeInfo returns the type information for a specified index. The index is previously obtained by a call to the GetTypeInfoCount method. If the method returns S_OK, the `TypeInfo` parameter is set to point to the requested type information; otherwise an error code is returned.

See Also

GetTypeInfoCount

GetIDsOfNames

Invoke

48.1.3.3.5 TEventDispatch.GetTypeInfoCount Method

GetTypeInfoCount returns the count of type information interfaces.

Pascal

```
function GetTypeInfoCount(out Count: Integer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetTypeInfoCount(int Count);
```

Description

GetTypeInfoCount returns the count of type information interfaces supported by the implementing COM object. If the method returns S_OK, the `Count` parameter is set to the number of type information interfaces, which is either 1 (if the object supports type information) or 0 (if the object does not support type information). A return value of E_NOTIMPL means that the method is not implemented in the COM object.

See Also

GetTypeInfo

GetIDsOfNames

Invoke

48.1.3.3.6 TEventDispatch.Invoke Method

Invoke performs a call to a late-binded property or method.

Pascal

```
function Invoke(DispID: Integer; const IID: TGUID; LocaleID: Integer; Flags: Word; var Params;
VarResult: Pointer; ExcepInfo: Pointer; ArgErr: Pointer): HResult; virtual; stdcall;
```

C++

```
virtual __fastcall __stdcall HRESULT Invoke(int DispID, const TGUID IID, int LocaleID, Word
Flags,  Params, void * VarResult, void * ExcepInfo, void * ArgErr);
```

Description

Invoke actually calls a method or a property exposed by the COM object through IDispatch. Invoke requires a previously obtained dispatch identifier. The identifier uniquely identifies a method, a property, and a list of parameters to be passed to the called method and exception information. The dispatch identifier of a method or property (also known as **dispid**) can be obtained by a calling the GetIDsOfNames method.

See Also

GetTypeInfoCount

GetTypeInfo

GetIDsOfNames

48.1.3.3.7 TEventDispatch.QueryInterface Method

Pascal

```
function QueryInterface(const IID: TGUID; out Obj): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT QueryInterface(const TGUID IID,  Obj);
```

Description

This is QueryInterface, a member of class TEventDispatch.

48.1.3.4 TEventDispatch Properties

The properties of the TEventDispatch class are listed here.

Protected Properties

	Name	Description
	Control (see page 2638)	This is Control, a member of class TEventDispatch.

48.1.3.4.1 TEventDispatch.Control Property

Pascal

```
property Control: TOleControl;
```

C++

```
__property TOleControl Control;
```

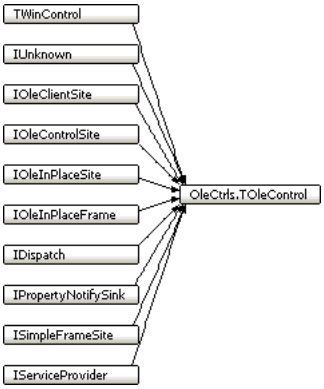
Description

This is Control, a member of class TEventDispatch.

48.1.4 TOleControl Class

TOleControl is the base class from which ActiveX controls are derived.

Class Hierarchy



Pascal

```
TOleControl = class(TWinControl, IUnknown, IOleClientSite, IOleControlSite, IOleInPlaceSite,
IOleInPlaceFrame, IDispatch, IPropertyNotifySink, ISimpleFrameSite, IServiceProvider);
```

C++

```
class TOleControl : public TWinControl, public IUnknown, public IOleClientSite, public
IOleControlSite, public IOleInPlaceSite, public IOleInPlaceFrame, public IDispatch, public
IPropertyNotifySink, public ISimpleFrameSite, public IServiceProvider;
```

File

OleCtrls

Description

TOleControl is derived from TWinControl and handles the interactions with COM necessary for using ActiveX controls. Most application do not need to explicitly derive a component from this class.

See Also

TOleContainer (see page 2529)

Code Generated When You Import Type Library Information

Using Component Wrappers

48.1.4.1 TOleControl Members

The following tables list the members exposed by TOleControl.

Protected Methods

	Name	Description
	_AddRef (see page 2646)	Called when an application uses a component interface.
	_Release (see page 2647)	Called when an application releases a component interface.
	_SetColorProp (see page 2647)	Sets a property of an ActiveX control to the specified color.
	_SetTColorProp (see page 2648)	Sets a property of an ActiveX control to the specified color.
	_SetTFontProp (see page 2648)	Sets a property of an ActiveX control to the specified font.
	_SetTOleEnumProp (see page 2649)	Sets a property of an ActiveX control to the specified enumeration value.
	_SetTPictureProp (see page 2649)	Sets a property of an ActiveX control to the specified picture object.
	CanInPlaceActivate (see page 2650)	Checks whether the ActiveX control supports in-place activation.
	ContextSensitiveHelp (see page 2650)	Checks whether the ActiveX control supports context-sensitive help.

	CreateWnd (see page 2651)	Creates the underlying window.
	DeactivateAndUndo (see page 2651)	Deactivates the in-place session and reverts the changes.
	DefineProperties (see page 2652)	Provides methods to read and write the IsControl property to a stream such as a form file.
	DestroyWindowHandle (see page 2652)	Destroys the window created in the CreateWindowHandle method.
	DiscardUndoState (see page 2653)	Discards the undo queue for the ActiveX control.
	EnableModeless (see page 2653)	Enables or disables the modeless dialogs of the ActiveX control.
	GetBorder (see page 2654)	Returns the rectangle in which the ActiveX control can draw toolbars.
	GetByteProp (see page 2654)	Reads a property of an ActiveX control. The property is expected to be of type `Byte`.
	GetColorProp (see page 2655)	Reads a property of an ActiveX control. The property is expected to be of type TColor.
	GetCompProp (see page 2655)	Reads a property of an ActiveX control. The property is expected to be of type `TColor`.
	GetContainer (see page 2655)	Returns a pointer to an `IOleClientSite` OLE interface.
	GetCurrencyProp (see page 2656)	Reads a property of an ActiveX control. The property is expected to be of type `Currency`.
	GetDoubleProp (see page 2656)	Reads a property of an ActiveX control. The property is expected to be of type `Double`.
	GetExtendedControl (see page 2657)	This is GetExtendedControl, a member of class TOleControl.
	GetIDispatchProp (see page 2657)	Reads a property of an ActiveX control. The property is expected to be of type IDispatch.
	GetIDsOfNames (see page 2658)	Maps a single member and an optional set of argument names to a corresponding set of integer dispatch identifiers (dispIDs).
	GetIntegerProp (see page 2658)	Reads a property of an ActiveX control. The property is expected to be of type `Integer`.
	GetIUnknownProp (see page 2658)	Reads a property of an ActiveX control. The property is expected to be of type IUnknown.
	GetMoniker (see page 2659)	This is GetMoniker, a member of class TOleControl.
	GetOleBoolProp (see page 2659)	Reads a property of an ActiveX control. The property is expected to be of type `TOleBool`.
	GetOleDateProp (see page 2660)	Reads a property of an ActiveX control. The property is expected to be of type `TOleDate`.
	GetOleEnumProp (see page 2660)	Reads a property of an ActiveX control. The property is expected to be of type `TOleEnum` (see page 2705).
	GetOleVariantProp (see page 2660)	Reads a property of an ActiveX control. The property is expected to be of type `OleVariant`.
	GetProperty (see page 2661)	Reads a property of an ActiveX control.
	GetShortIntProp (see page 2662)	Reads a property of an ActiveX control. The property is expected to be of type `ShortInt`.
	GetSingleProp (see page 2662)	Reads a property of an ActiveX control. The property is expected to be of type `Single`.
	GetSmallintProp (see page 2662)	Reads a property of an ActiveX control. The property is expected to be of type `Smallint`.
	GetStringProp (see page 2663)	Reads a property of an ActiveX control. The property is expected to be of type `String`.
	GetTColorProp (see page 2663)	Reads a property of an ActiveX control. The property is expected to be of type TColor.
	GetTDateTimeProp (see page 2664)	Reads a property of an ActiveX control. The property is expected to be of type TDateTime.
	GetTFontProp (see page 2664)	Reads a property of an ActiveX control. The property is expected to be of type TFont.
	GetTOleEnumProp (see page 2664)	Reads a property of an ActiveX control. The property is expected to be of type `TOleEnum` (see page 2705).
	GetTPictureProp (see page 2665)	Reads a property of an ActiveX control. The property is expected to be of type TPicture.
	GetTypeInfo (see page 2665)	Retrieves the type information for an object.
	GetTypeInfoCount (see page 2665)	Returns the number of type information interfaces that an object provides (either 0 or 1).
	GetVariantProp (see page 2666)	Reads a property of an ActiveX control.
	GetWideStringProp (see page 2666)	Reads a property of an ActiveX control. The property is expected to be of type `WideString`.

	GetWindowContext (see page 2667)	This is GetWindowContext, a member of class TOleControl.
	GetWordBoolProp (see page 2667)	Reads a property of an ActiveX control. The property is expected to be of type `WordBool`.
	GetWordProp (see page 2667)	Reads a property of an ActiveX control. The property is expected to be of type `Word`.
A	InitControlData (see page 2668)	This is InitControlData, a member of class TOleControl.
V	InitControlInterface (see page 2668)	This is InitControlInterface, a member of class TOleControl.
	InsertMenus (see page 2668)	This is InsertMenus, a member of class TOleControl.
	Invoke (see page 2668)	Provides access to Automation properties and methods when the component wraps an Automation object.
V	InvokeEvent (see page 2669)	This is InvokeEvent, a member of class TOleControl.
	InvokeMethod (see page 2669)	This is InvokeMethod, a member of class TOleControl.
	LockInPlaceActive (see page 2670)	This is LockInPlaceActive, a member of class TOleControl.
	OleControlSite_TranslateAccelerator (see page 2670)	This is OleControlSite_TranslateAccelerator, a member of class TOleControl.
	OleInPlaceFrame_GetWindow (see page 2670)	This is OleInPlaceFrame_GetWindow, a member of class TOleControl.
	OleInPlaceFrame_TranslateAccelerator (see page 2670)	This is OleInPlaceFrame_TranslateAccelerator, a member of class TOleControl.
	OleInPlaceSite_GetWindow (see page 2671)	This is OleInPlaceSite_GetWindow, a member of class TOleControl.
V	OnChanged (see page 2671)	This is OnChanged, a member of class TOleControl.
	OnControlInfoChanged (see page 2671)	This is OnControlInfoChanged, a member of class TOleControl.
	OnFocus (see page 2671)	This is OnFocus, a member of class TOleControl.
	OnInPlaceActivate (see page 2672)	This is OnInPlaceActivate, a member of class TOleControl.
	OnInPlaceDeactivate (see page 2672)	This is OnInPlaceDeactivate, a member of class TOleControl.
	OnPosRectChange (see page 2672)	This is OnPosRectChange, a member of class TOleControl.
V	OnRequestEdit (see page 2672)	This is OnRequestEdit, a member of class TOleControl.
	OnShowWindow (see page 2672)	This is OnShowWindow, a member of class TOleControl.
	OnUIActivate (see page 2673)	This is OnUIActivate, a member of class TOleControl.
	OnUIDeactivate (see page 2673)	This is OnUIDeactivate, a member of class TOleControl.
V	PaletteChanged (see page 2673)	Responds to changes in the system's palette by realizing the control's palette and the palette for each child control.
	PictureChanged (see page 2674)	This is PictureChanged, a member of class TOleControl.
	PostMessageFilter (see page 2674)	This is PostMessageFilter, a member of class TOleControl.
	PreMessageFilter (see page 2674)	This is PreMessageFilter, a member of class TOleControl.
V	QueryInterface (see page 2675)	Returns a reference to a specified interface if the object supports that interface.
	QueryService (see page 2675)	This is QueryService, a member of class TOleControl.
	RemoveMenus (see page 2675)	This is RemoveMenus, a member of class TOleControl.
	RequestBorderSpace (see page 2675)	This is RequestBorderSpace, a member of class TOleControl.
	RequestNewObjectLayout (see page 2676)	This is RequestNewObjectLayout, a member of class TOleControl.
	SaveObject (see page 2676)	Saves the object to persistent storage.
	Scroll (see page 2676)	Scrolls the view by the indicated amount of pixels.
	SetActiveObject (see page 2677)	This is SetActiveObject, a member of class TOleControl.
	SetBorderSpace (see page 2677)	Allocates border space for the ActiveX control.
	SetByteProp (see page 2677)	Sets a property of an ActiveX control to the specified value of type `Byte`.
	SetColorProp (see page 2678)	Sets a property of an ActiveX control to the specified color.
	SetCompProp (see page 2678)	Sets a property of an ActiveX control to the specified value of type `Comp`.
	SetCurrencyProp (see page 2679)	Sets a property of an ActiveX control to the specified value of type `Currency`.
	SetDoubleProp (see page 2679)	Sets a property of an ActiveX control to the specified value of type `Double`.
	SetIDispatchProp (see page 2679)	Sets a property of an ActiveX control to the specified value of type IDispatch.
	SetIntegerProp (see page 2680)	Sets a property of an ActiveX control to the specified value of type `Integer`.
	SetIUnknownProp (see page 2680)	Sets a property of an ActiveX control to the specified value of type IUnknown.
	SetMenu (see page 2680)	Attaches a menu to the in-place activated ActiveX object.
V	SetName (see page 2681)	Sets the name of the control.
	SetOleBoolProp (see page 2681)	Sets a property of an ActiveX control to the specified value of type `TOleBool`.
	SetOleDateProp (see page 2681)	Sets a property of an ActiveX control to the specified value of type `TOleDate`.
	SetOleEnumProp (see page 2682)	Sets a property of an ActiveX control to the specified value of type TOleEnum (see page 2705).
	SetOleVariantProp (see page 2682)	Sets a property of an ActiveX control to the specified value of type `OleVariant`.

	SetParent (see page 2682)	Sets the parent of the control.
	SetProperty (see page 2683)	Sets a property of an ActiveX control to the specified value.
	SetShortIntProp (see page 2684)	Sets a property of an ActiveX control to the specified value of type `ShortInt`.
	SetSingleProp (see page 2684)	Sets a property of an ActiveX control to the specified value of type `Single`.
	SetSmallintProp (see page 2684)	Sets a property of an ActiveX control to the specified value of type `Smallint`.
	SetStatusText (see page 2685)	This is SetStatusText, a member of class TOleControl.
	SetStringProp (see page 2685)	Sets a property of an ActiveX control to the specified value of type `String`.
	SetTColorProp (see page 2685)	Sets a property of an ActiveX control to the specified color.
	SetTDateTimeProp (see page 2686)	Sets a property of an ActiveX control to the specified value of type TDateTime.
	SetTFontProp (see page 2686)	Sets a property of an ActiveX control to the specified font.
	SetTOleEnumProp (see page 2686)	Sets a property of an ActiveX control to the specified enumeration value.
	SetTPictureProp (see page 2687)	Sets a property of an ActiveX control to the specified picture object.
	SetVariantProp (see page 2687)	Sets a property of an ActiveX control to the specified value.
	SetWideStringProp (see page 2688)	Sets a property of an ActiveX control to the specified value of type `WideString`.
	SetWordBoolProp (see page 2688)	Sets a property of an ActiveX control to the specified value of type `WordBool`.
	SetWordProp (see page 2688)	Sets a property of an ActiveX control to the specified value of type `Word`.
	ShowObject (see page 2689)	Makes the enclosed object visible.
	ShowPropertyFrame (see page 2689)	Specifies whether the TOleControl (see page 2638) shows a property sheet.
	StandardEvent (see page 2690)	This is StandardEvent, a member of class TOleControl.
	SuppressException (see page 2690)	SuppressException suppresses OLE exceptions at design time.
	TransformCoords (see page 2690)	Translates a set of coordinates.
	WndProc (see page 2691)	Provides specific message responses for the control.

Protected Properties

	Name	Description
	ControlData (see page 2696)	Specifies the PControlData (see page 2704) associated with TOleControl (see page 2638).

Public Events

	Name	Description
	ServiceQuery (see page 2698)	Specifies the TServiceQuery (see page 2706) associated with this ActiveX control.

Public Methods

	Name	Description
	BrowseProperties (see page 2649)	Notifies the ActiveX control to display its Properties dialog box.
	Create (see page 2643)	Creates a new instance of TOleControl (see page 2638).
	DefaultHandler (see page 2651)	Handles all otherwise-unhandled messages.
	Destroy (see page 2643)	Frees the memory associated with this instance of TOleControl (see page 2638).
	DoObjectVerb (see page 2653)	Notifies the ActiveX control to perform one of its actions.
	GetEnumPropDesc (see page 2656)	Returns descriptions of the ActiveX control's properties.
	GetHelpContext (see page 2657)	Retrieves the Help context of a member of the ActiveX control.
	GetObjectVerbs (see page 2659)	Retrieves the names of the verbs implemented by the ActiveX control.
	GetPropDisplayString (see page 2661)	Returns the display string for an interface for the ActiveX control.
	GetPropDisplayStrings (see page 2661)	Returns the display strings for an interface for the ActiveX control.
	IsCustomProperty (see page 2669)	Indicates whether an interface for the ActiveX control is a custom property.
	IsPropPageProperty (see page 2669)	Indicates whether an interface for the OCX control is a property page property.
	SetBounds (see page 2677)	Sets the boundaries of the ActiveX control.
	SetPropDisplayString (see page 2683)	Sets the display string for an interface for the ActiveX control.
	ShowAboutBox (see page 2689)	Notifies the ActiveX control to display its About box.

Public Properties

	Name	Description
	Anchors (see page 2695)	Specifies how the control is anchored to its parent.
	DefaultDispatch (see page 2696)	The default dispatch interface.

	OleObject (see page 2697)	The ActiveX control associated with the TOleControl (see page 2638) object.
	PerPropBrowsing (see page 2697)	Provides access to information about the property page properties of the OleControl.
	TabStop (see page 2697)	Determines if the user can tab to a control.

48.1.4.2 TOleControl.Create Constructor

Creates a new instance of TOleControl (see page 2638).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TOleControl(TComponent * AOwner);
```

Description

Use Create to create a new instance of the TOleControl (see page 2638) object. Create calls the inherited Create method and creates and initializes internal objects.

See Also

Destroy (see page 2643)

Create

48.1.4.3 TOleControl.Destroy Destructor

Frees the memory associated with this instance of TOleControl (see page 2638).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TOleControl();
```

Description

Do not call Destroy. Instead, call Free, which calls Destroy if this instance of TOleControl (see page 2638) is not nil.

See Also

Create (see page 2643)

System::TObject::Free

48.1.4.4 TOleControl Methods

The methods of the TOleControl class are listed here.

Protected Methods

	Name	Description
	_AddRef (see page 2646)	Called when an application uses a component interface.
	_Release (see page 2647)	Called when an application releases a component interface.
	_SetColorProp (see page 2647)	Sets a property of an ActiveX control to the specified color.
	_SetTColorProp (see page 2648)	Sets a property of an ActiveX control to the specified color.

	_SetTFontProp (see page 2648)	Sets a property of an ActiveX control to the specified font.
	_SetTOleEnumProp (see page 2649)	Sets a property of an ActiveX control to the specified enumeration value.
	_SetTPictureProp (see page 2649)	Sets a property of an ActiveX control to the specified picture object.
	CanInPlaceActivate (see page 2650)	Checks whether the ActiveX control supports in-place activation.
	ContextSensitiveHelp (see page 2650)	Checks whether the ActiveX control supports context-sensitive help.
V	CreateWnd (see page 2651)	Creates the underlying window.
	DeactivateAndUndo (see page 2651)	Deactivates the in-place session and reverts the changes.
V	DefineProperties (see page 2652)	Provides methods to read and write the IsControl property to a stream such as a form file.
V	DestroyWindowHandle (see page 2652)	Destroys the window created in the CreateWindowHandle method.
	DiscardUndoState (see page 2653)	Discards the undo queue for the ActiveX control.
	EnableModeless (see page 2653)	Enables or disables the modeless dialogs of the ActiveX control.
	GetBorder (see page 2654)	Returns the rectangle in which the ActiveX control can draw toolbars.
	GetByteProp (see page 2654)	Reads a property of an ActiveX control. The property is expected to be of type `Byte`.
	GetColorProp (see page 2655)	Reads a property of an ActiveX control. The property is expected to be of type TColor.
	GetCompProp (see page 2655)	Reads a property of an ActiveX control. The property is expected to be of type `TColor`.
	GetContainer (see page 2655)	Returns a pointer to an `IOleClientSite` OLE interface.
	GetCurrencyProp (see page 2656)	Reads a property of an ActiveX control. The property is expected to be of type `Currency`.
	GetDoubleProp (see page 2656)	Reads a property of an ActiveX control. The property is expected to be of type `Double`.
	GetExtendedControl (see page 2657)	This is GetExtendedControl, a member of class TOleControl.
	GetIDispatchProp (see page 2657)	Reads a property of an ActiveX control. The property is expected to be of type IDispatch.
	GetIDsOfNames (see page 2658)	Maps a single member and an optional set of argument names to a corresponding set of integer dispatch identifiers (dispIDs).
	GetIntegerProp (see page 2658)	Reads a property of an ActiveX control. The property is expected to be of type `Integer`.
	GetIUnknownProp (see page 2658)	Reads a property of an ActiveX control. The property is expected to be of type IUnknown.
	GetMoniker (see page 2659)	This is GetMoniker, a member of class TOleControl.
	GetOleBoolProp (see page 2659)	Reads a property of an ActiveX control. The property is expected to be of type `TOleBool`.
	GetOleDateProp (see page 2660)	Reads a property of an ActiveX control. The property is expected to be of type `TOleDate`.
	GetOleEnumProp (see page 2660)	Reads a property of an ActiveX control. The property is expected to be of type `TOleEnum` (see page 2705).
	GetOleVariantProp (see page 2660)	Reads a property of an ActiveX control. The property is expected to be of type `OleVariant`.
	GetProperty (see page 2661)	Reads a property of an ActiveX control.
	GetShortIntProp (see page 2662)	Reads a property of an ActiveX control. The property is expected to be of type `ShortInt`.
	GetSingleProp (see page 2662)	Reads a property of an ActiveX control. The property is expected to be of type `Single`.
	GetSmallintProp (see page 2662)	Reads a property of an ActiveX control. The property is expected to be of type `Smallint`.
	GetStringProp (see page 2663)	Reads a property of an ActiveX control. The property is expected to be of type `String`.
	GetTColorProp (see page 2663)	Reads a property of an ActiveX control. The property is expected to be of type TColor.
	GetTDateTimeProp (see page 2664)	Reads a property of an ActiveX control. The property is expected to be of type TDateTime.
	GetTFontProp (see page 2664)	Reads a property of an ActiveX control. The property is expected to be of type TFont.
	GetTOleEnumProp (see page 2664)	Reads a property of an ActiveX control. The property is expected to be of type `TOleEnum` (see page 2705).
	GetTPictureProp (see page 2665)	Reads a property of an ActiveX control. The property is expected to be of type TPicture.

	GetTypeInfo (see page 2665)	Retrieves the type information for an object.
	GetTypeInfoCount (see page 2665)	Returns the number of type information interfaces that an object provides (either 0 or 1).
	GetVariantProp (see page 2666)	Reads a property of an ActiveX control.
	GetWideStringProp (see page 2666)	Reads a property of an ActiveX control. The property is expected to be of type `WideString`.
	GetWindowContext (see page 2667)	This is GetWindowContext, a member of class TOleControl.
	GetWordBoolProp (see page 2667)	Reads a property of an ActiveX control. The property is expected to be of type `WordBool`.
	GetWordProp (see page 2667)	Reads a property of an ActiveX control. The property is expected to be of type `Word`.
A	InitControlData (see page 2668)	This is InitControlData, a member of class TOleControl.
V	InitControlInterface (see page 2668)	This is InitControlInterface, a member of class TOleControl.
	InsertMenus (see page 2668)	This is InsertMenus, a member of class TOleControl.
	Invoke (see page 2668)	Provides access to Automation properties and methods when the component wraps an Automation object.
V	InvokeEvent (see page 2669)	This is InvokeEvent, a member of class TOleControl.
	InvokeMethod (see page 2669)	This is InvokeMethod, a member of class TOleControl.
	LockInPlaceActive (see page 2670)	This is LockInPlaceActive, a member of class TOleControl.
	OleControlSite_TranslateAccelerator (see page 2670)	This is OleControlSite_TranslateAccelerator, a member of class TOleControl.
	OleInPlaceFrame_GetWindow (see page 2670)	This is OleInPlaceFrame_GetWindow, a member of class TOleControl.
	OleInPlaceFrame_TranslateAccelerator (see page 2670)	This is OleInPlaceFrame_TranslateAccelerator, a member of class TOleControl.
	OleInPlaceSite_GetWindow (see page 2671)	This is OleInPlaceSite_GetWindow, a member of class TOleControl.
V	OnChanged (see page 2671)	This is OnChanged, a member of class TOleControl.
	OnControlInfoChanged (see page 2671)	This is OnControlInfoChanged, a member of class TOleControl.
	OnFocus (see page 2671)	This is OnFocus, a member of class TOleControl.
	OnInPlaceActivate (see page 2672)	This is OnInPlaceActivate, a member of class TOleControl.
	OnInPlaceDeactivate (see page 2672)	This is OnInPlaceDeactivate, a member of class TOleControl.
	OnPosRectChange (see page 2672)	This is OnPosRectChange, a member of class TOleControl.
V	OnRequestEdit (see page 2672)	This is OnRequestEdit, a member of class TOleControl.
	OnShowWindow (see page 2672)	This is OnShowWindow, a member of class TOleControl.
	OnUIActivate (see page 2673)	This is OnUIActivate, a member of class TOleControl.
	OnUIDeactivate (see page 2673)	This is OnUIDeactivate, a member of class TOleControl.
V	PaletteChanged (see page 2673)	Responds to changes in the system's palette by realizing the control's palette and the palette for each child control.
	PictureChanged (see page 2674)	This is PictureChanged, a member of class TOleControl.
	PostMessageFilter (see page 2674)	This is PostMessageFilter, a member of class TOleControl.
	PreMessageFilter (see page 2674)	This is PreMessageFilter, a member of class TOleControl.
V	QueryInterface (see page 2675)	Returns a reference to a specified interface if the object supports that interface.
	QueryService (see page 2675)	This is QueryService, a member of class TOleControl.
	RemoveMenus (see page 2675)	This is RemoveMenus, a member of class TOleControl.
	RequestBorderSpace (see page 2675)	This is RequestBorderSpace, a member of class TOleControl.
	RequestNewObjectLayout (see page 2676)	This is RequestNewObjectLayout, a member of class TOleControl.
	SaveObject (see page 2676)	Saves the object to persistent storage.
	Scroll (see page 2676)	Scrolls the view by the indicated amount of pixels.
	SetActiveObject (see page 2677)	This is SetActiveObject, a member of class TOleControl.
	SetBorderSpace (see page 2677)	Allocates border space for the ActiveX control.
	SetByteProp (see page 2677)	Sets a property of an ActiveX control to the specified value of type `Byte`.
	SetColorProp (see page 2678)	Sets a property of an ActiveX control to the specified color.
	SetCompProp (see page 2678)	Sets a property of an ActiveX control to the specified value of type `Comp`.
	SetCurrencyProp (see page 2679)	Sets a property of an ActiveX control to the specified value of type `Currency`.
	SetDoubleProp (see page 2679)	Sets a property of an ActiveX control to the specified value of type `Double`.
	SetIDispatchProp (see page 2679)	Sets a property of an ActiveX control to the specified value of type IDispatch.
	SetIntegerProp (see page 2680)	Sets a property of an ActiveX control to the specified value of type `Integer`.
	SetIUnknownProp (see page 2680)	Sets a property of an ActiveX control to the specified value of type IUnknown.
	SetMenu (see page 2680)	Attaches a menu to the in-place activated ActiveX object.

	SetName (see page 2681)	Sets the name of the control.
	SetOleBoolProp (see page 2681)	Sets a property of an ActiveX control to the specified value of type `TOleBool`.
	SetOleDateProp (see page 2681)	Sets a property of an ActiveX control to the specified value of type `TOleDate`.
	SetOleEnumProp (see page 2682)	Sets a property of an ActiveX control to the specified value of type TOleEnum (see page 2705).
	SetOleVariantProp (see page 2682)	Sets a property of an ActiveX control to the specified value of type `OleVariant`.
	SetParent (see page 2682)	Sets the parent of the control.
	SetProperty (see page 2683)	Sets a property of an ActiveX control to the specified value.
	SetShortIntProp (see page 2684)	Sets a property of an ActiveX control to the specified value of type `ShortInt`.
	SetSingleProp (see page 2684)	Sets a property of an ActiveX control to the specified value of type `Single`.
	SetSmallintProp (see page 2684)	Sets a property of an ActiveX control to the specified value of type `Smallint`.
	SetStatusText (see page 2685)	This is SetStatusText, a member of class TOleControl.
	SetStringProp (see page 2685)	Sets a property of an ActiveX control to the specified value of type `String`.
	SetTColorProp (see page 2685)	Sets a property of an ActiveX control to the specified color.
	SetTDateTimeProp (see page 2686)	Sets a property of an ActiveX control to the specified value of type TDateTime.
	SetTFontProp (see page 2686)	Sets a property of an ActiveX control to the specified font.
	SetTOleEnumProp (see page 2686)	Sets a property of an ActiveX control to the specified enumeration value.
	SetTPictureProp (see page 2687)	Sets a property of an ActiveX control to the specified picture object.
	SetVariantProp (see page 2687)	Sets a property of an ActiveX control to the specified value.
	SetWideStringProp (see page 2688)	Sets a property of an ActiveX control to the specified value of type `WideString`.
	SetWordBoolProp (see page 2688)	Sets a property of an ActiveX control to the specified value of type `WordBool`.
	SetWordProp (see page 2688)	Sets a property of an ActiveX control to the specified value of type `Word`.
	ShowObject (see page 2689)	Makes the enclosed object visible.
	ShowPropertyFrame (see page 2689)	Specifies whether the TOleControl (see page 2638) shows a property sheet.
	StandardEvent (see page 2690)	This is StandardEvent, a member of class TOleControl.
	SuppressException (see page 2690)	SuppressException suppresses OLE exceptions at design time.
	TransformCoords (see page 2690)	Translates a set of coordinates.
	WndProc (see page 2691)	Provides specific message responses for the control.

Public Methods

	Name	Description
	BrowseProperties (see page 2649)	Notifies the ActiveX control to display its Properties dialog box.
	DefaultHandler (see page 2651)	Handles all otherwise-unhandled messages.
	DoObjectVerb (see page 2653)	Notifies the ActiveX control to perform one of its actions.
	GetEnumPropDesc (see page 2656)	Returns descriptions of the ActiveX control's properties.
	GetHelpContext (see page 2657)	Retrieves the Help context of a member of the ActiveX control.
	GetObjectVerbs (see page 2659)	Retrieves the names of the verbs implemented by the ActiveX control.
	GetPropDisplayString (see page 2661)	Returns the display string for an interface for the ActiveX control.
	GetPropDisplayStrings (see page 2661)	Returns the display strings for an interface for the ActiveX control.
	IsCustomProperty (see page 2669)	Indicates whether an interface for the ActiveX control is a custom property.
	IsPropPageProperty (see page 2669)	Indicates whether an interface for the OCX control is a property page property.
	SetBounds (see page 2677)	Sets the boundaries of the ActiveX control.
	SetPropDisplayString (see page 2683)	Sets the display string for an interface for the ActiveX control.
	ShowAboutBox (see page 2689)	Notifies the ActiveX control to display its About box.

48.1.4.4.1 TOleControl._AddRef Method

Called when an application uses a component interface.

Pascal

```
function _AddRef: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _AddRef();
```

Description

__AddRef is a basic implementation of the IInterface method, _AddRef.

Note: If the component is a wrapper for a COM object, __AddRef calls the _AddRef method of that COM object, and returns the resulting reference count.

In all other cases, __AddRef simply returns –1 and takes no action. This allows the component to implement interfaces where reference counting is not required. More sophisticated components should override __AddRef to implement reference counting.

See Also

QueryInterface

_Release

_AddRef

48.1.4.4.2 TOleControl._Release Method

Called when an application releases a component interface.

Pascal

```
function _Release: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _Release();
```

Description

__Release is a basic implementation of the IInterface method, _Release.

__Release returns the resulting value of the reference count for the component's interface.

Note:

In all other cases, __Release simply returns –1 and takes no action. This allows the component to implement interfaces where reference counting is not required. More sophisticated components should override __Release to implement reference counting.

See Also

QueryInterface

_AddRef

48.1.4.4.3 TOleControl._SetColorProp Method

Sets a property of an ActiveX control to the specified color.

Pascal

```
procedure _SetColorProp(Index: Integer; Value: TColor);
```

C++

```
__fastcall _SetColorProp(int Index, TColor Value);
```

Description

Use _SetColorProp to set a property of an ActiveX control to the specified color. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set color.

Note: _SetColorProp simply calls the SetColorProp (see page 2678) method.

See Also

SetColorProp (see page 2678)

_SetTColorProp (see page 2648)

SetTColorProp (see page 2685)

48.1.4.4.4 TOleControl._SetTColorProp Method

Sets a property of an ActiveX control to the specified color.

Pascal

```
procedure _SetTColorProp(Index: Integer; Value: TColor);
```

C++

```
__fastcall _SetTColorProp(int Index, TColor Value);
```

Description

Use _SetTColorProp to set a property of an ActiveX control to the specified color. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set color.

Note: _SetTColorProp simply calls the SetTColorProp (see page 2685) method.

See Also

SetTColorProp (see page 2685)

_SetColorProp (see page 2647)

SetColorProp (see page 2678)

48.1.4.4.5 TOleControl._SetTFontProp Method

Sets a property of an ActiveX control to the specified font.

Pascal

```
procedure _SetTFontProp(Index: Integer; Value: TFont);
```

C++

```
__fastcall _SetTFontProp(int Index, TFont Value);
```

Description

Use _SetTFontProp to set a property of an ActiveX control to the specified font. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set font.

Note: _SetTFontProp simply calls the SetTFontProp (see page 2686) method.

See Also

SetTFontProp (see page 2686)

SetTColorProp (see page 2685)

SetTPictureProp (see page 2687)

SetTOleEnumProp (see page 2686)

48.1.4.4.6 TOleControl._SetTOleEnumProp Method

Sets a property of an ActiveX control to the specified enumeration value.

Pascal

```
procedure _SetTOleEnumProp(Index: Integer; Value: TOleEnum);
```

C++

```
__fastcall _SetTOleEnumProp(int Index, TOleEnum Value);
```

Description

Use _SetTOleEnumProp to set a property of an ActiveX control to the specified enumeration value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set enumeration value.

Note: _SetTOleEnumProp simply calls the SetTOleEnumProp (see page 2686) method.

See Also

SetTOleEnumProp (see page 2686)

SetTFontProp (see page 2686)

SetTColorProp (see page 2685)

SetTPictureProp (see page 2687)

48.1.4.4.7 TOleControl._SetTPictureProp Method

Sets a property of an ActiveX control to the specified picture object.

Pascal

```
procedure _SetTPictureProp(Index: Integer; Value: TPicture);
```

C++

```
__fastcall _SetTPictureProp(int Index, TPicture Value);
```

Description

Use _SetTPictureProp to set a property of an ActiveX control to the specified picture object. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set picture object.

Note: _SetTPictureProp simply calls the SetTPictureProp (see page 2687) method.

See Also

SetTPictureProp (see page 2687)

SetTColorProp (see page 2685)

SetTFontProp (see page 2686)

SetTOleEnumProp (see page 2686)

48.1.4.4.8 TOleControl.BrowseProperties Method

Notifies the ActiveX control to display its Properties dialog box.

Pascal

```
procedure BrowseProperties;
```

C++

```
__fastcall BrowseProperties();
```

Description

Use BrowseProperties to notify the ActiveX control to display its Properties dialog box. The Properties dialog box contains property pages for the ActiveX control.

48.1.4.4.9 TOleControl.CanInPlaceActivate Method

Checks whether the ActiveX control supports in-place activation.

Pascal

```
function CanInPlaceActivate: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT CanInPlaceActivate();
```

Description

Call CanInPlaceActivate to determine whether the ActiveX control can be activated in-place. The container of the ActiveX control automatically calls the method when the container must activate the ActiveX control in-place.

Note: As implemented in TOleControl (see page 2638), CanInPlaceActivate always returns S_OK, which means that the ActiveX control can always be activated in-place.

See Also

OnInPlaceActivate (see page 2672)

OnInPlaceDeactivate (see page 2672)

OnUIActivate (see page 2673)

OnUIDeactivate (see page 2673)

48.1.4.4.10 TOleControl.ContextSensitiveHelp Method

Checks whether the ActiveX control supports context-sensitive help.

Pascal

```
function ContextSensitiveHelp(fEnterMode: BOOL): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT ContextSensitiveHelp(BOOL fEnterMode);
```

Description

Call ContextSensitiveHelp to determine whether the ActiveX control supports context-sensitive help when in-place mode is activated. This method is called automatically when the in-place mode is activated by the container of the ActiveX control.

Note: As implemented in TOleControl (see page 2638), ContextSensitiveHelp always returns S_OK, which means that the context-sensitive help is always enabled.

See Also

CanInPlaceActivate (see page 2650)

OnInPlaceActivate (see page 2672)

48.1.4.4.11 TOleControl.CreateWnd Method

Creates the underlying window.

Pascal

```
procedure CreateWnd; override;
```

C++

```
virtual __fastcall CreateWnd();
```

Description

CreateWnd is called automatically when the control is first created or when the underlying screen object must be destroyed and recreated to reflect property changes. Override CreateWnd to pass additional initialization messages to the screen object when it is created.

See Also

CreateParams

CreateWindowHandle

Perform

DestroyWnd

48.1.4.4.12 TOleControl.DeactivateAndUndo Method

Deactivates the in-place session and reverts the changes.

Pascal

```
function DeactivateAndUndo: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT DeactivateAndUndo();
```

Description

Call DeactivateAndUndo to deactivate the current in-place session of the ActiveX control. DeactivateAndUndo also notifies the control to revert to the internally stored undo state.

See Also

OnInPlaceDeactivate (see page 2672)

OnUIDeactivate (see page 2673)

48.1.4.4.13 TOleControl.DefaultHandler Method

Handles all otherwise-unhandled messages.

Pascal

```
procedure DefaultHandler(var Message); override;
```

C++

```
virtual __fastcall DefaultHandler( Message);
```

Description

DefaultHandler handles all messages not already handled in a control's WndProc (see page 2691) or message handling

methods. The inherited method should always be called for unhandled messages.

See Also

DefaultHandler

48.1.4.4.14 TOleControl.DefineProperties Method

Provides methods to read and write the IsControl property to a stream such as a form file.

Pascal

```
procedure DefineProperties(Filer: TFiler); override;
```

C++

```
virtual __fastcall DefineProperties(TFiler Filer);
```

Description

DefineProperties is called automatically by the streaming system that loads and saves VCL components. It allows persistent objects to read and write properties that are not published. Published properties are loaded and stored automatically.

DefineProperties uses the methods of the filer passed as the Filer parameter to assign methods that can load and save the IsControl property.

Override DefineProperties to persist unpublished properties. Call the DefineBinaryProperty or DefineProperty method of the Filer parameter to assign methods that load or save property values.

Note: When overriding this method, call the parent class's DefineProperties method first.

See Also

DefineBinaryProperty

DefineProperty

DefineProperties

IsControl

48.1.4.4.15 TOleControl.DestroyWindowHandle Method

Destroys the window created in the CreateWindowHandle method.

Pascal

```
procedure DestroyWindowHandle; override;
```

C++

```
virtual __fastcall DestroyWindowHandle();
```

Description

Call DestroyWindowHandle to dispose of the window for the control.

TWinControl's Destroy method calls DestroyWindowHandle to destroy any window associated with a windowed control before destroying the object. The DestroyWnd method also calls DestroyWindowHandle.

See Also

CreateWindowHandle

Destroy

DestroyWnd

48.1.4.4.16 TOleControl.DiscardUndoState Method

Discards the undo queue for the ActiveX control.

Pascal

```
function DiscardUndoState: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT DiscardUndoState();
```

Description

Call DiscardUndoState to clear the undo queue of the ActiveX control contained within the container.

See Also

OnInPlaceDeactivate (see page 2672)

DeactivateAndUndo (see page 2651)

48.1.4.4.17 TOleControl.DoObjectVerb Method

Notifies the ActiveX control to perform one of its actions.

Pascal

```
procedure DoObjectVerb(Verb: Integer);
```

C++

```
__fastcall DoObjectVerb(int Verb);
```

Description

Use DoObjectVerb to notify the ActiveX control to perform one of its actions. Verb is an integer that corresponds to the action. Most application-defined verbs are small positive integers.

Verb Name	Verb
OLEIVERB_PRIMARY	0
OLEIVERB_SHOW	–1
OLEIVERB_OPEN	–2
OLEIVERB_HIDE	–3
OLEIVERB_UIACTIVATE	–4
OLEIVERB_INPLACEACTIVATE	–5
OLEIVERB_DISCARDUNDOSTATE	–6
OLEIVERB_PROPERTIES	–7

For more information about verbs, search the OLE.HLP file or the latest Microsoft Developer Network CD.

48.1.4.4.18 TOleControl.EnableModeless Method

Enables or disables the modeless dialogs of the ActiveX control.

Pascal

```
function EnableModeless(fEnable: BOOL): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT EnableModeless(BOOL fEnable);
```

Description

Call EnableModeless to enable or disable the modeless dialog boxes of the ActiveX control. To make modal dialogs work properly, you must perform a call to the EnableModeless methods with the `fEnable` parameter set to **False**.

See Also

CanInPlaceActivate (see page 2650)

OnInPlaceActivate (see page 2672)

GetBorder (see page 2654)

48.1.4.4.19 TOleControl.GetBorder Method

Returns the rectangle in which the ActiveX control can draw toolbars.

Pascal

```
function GetBorder(out rectBorder: TRect): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetBorder(TRect rectBorder);
```

Description

Call GetBorder to receive the rectangle in which the ActiveX control can draw custom controls, such as toolbars. The drawn controls are only visible when the container is in place mode.

See Also

CanInPlaceActivate (see page 2650)

OnInPlaceActivate (see page 2672)

EnableModeless (see page 2653)

48.1.4.4.20 TOleControl.GetByteProp Method

Reads a property of an ActiveX control. The property is expected to be of type `Byte`.

Pascal

```
function GetByteProp(Index: Integer): Byte;
```

C++

```
__fastcall Byte GetByteProp(int Index);
```

Description

Use GetByteProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetByteProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.21 TOleControl.GetColorProp Method

Reads a property of an ActiveX control. The property is expected to be of type TColor.

Pascal

```
function GetColorProp(Index: Integer): TColor;
```

C++

```
__fastcall TColor GetColorProp(int Index);
```

Description

Use GetColorProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetColorProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetTColorProp (see page 2663)

GetVariantProp (see page 2666)

48.1.4.4.22 TOleControl.GetCompProp Method

Reads a property of an ActiveX control. The property is expected to be of type `TColor`.

Pascal

```
function GetCompProp(Index: Integer): Comp;
```

C++

```
__fastcall Comp GetCompProp(int Index);
```

Description

Use GetCompProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetCompProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.23 TOleControl.GetContainer Method

Returns a pointer to an `IOleClientSite` OLE interface.

Pascal

```
function GetContainer(out container: IOleContainer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetContainer(IOleContainer container);
```

Description

Use GetContainer to obtain a pointer to an `IOleClientSite` OLE interface implemented by this TOleControl (see page 2638).

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.24 TOleControl.GetCurrencyProp Method

Reads a property of an ActiveX control. The property is expected to be of type `Currency`.

Pascal

```
function GetCurrencyProp(Index: Integer): Currency;
```

C++

```
__fastcall Currency GetCurrencyProp(int Index);
```

Description

Use GetCurrencyProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetCurrencyProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.25 TOleControl.GetDoubleProp Method

Reads a property of an ActiveX control. The property is expected to be of type `Double`.

Pascal

```
function GetDoubleProp(Index: Integer): Double;
```

C++

```
__fastcall Double GetDoubleProp(int Index);
```

Description

Use GetDoubleProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetDoubleProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.26 TOleControl.GetEnumPropDesc Method

Returns descriptions of the ActiveX control's properties.

Pascal

```
function GetEnumPropDesc(DispID: Integer): TEnumPropDesc;
```

C++

```
__fastcall TEnumPropDesc GetEnumPropDesc(int DispID);
```

Description

Use GetEnumPropDesc to return descriptions of the ActiveX control's properties. GetEnumPropDesc returns the descriptions in a TEnumPropDesc (see page 2631) object.

48.1.4.4.27 TOleControl.GetExtendedControl Method

Pascal

```
function GetExtendedControl(out disp: IDispatch): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetExtendedControl(IDispatch disp);
```

Description

This is GetExtendedControl, a member of class TOleControl.

48.1.4.4.28 TOleControl.GetHelpContext Method

Retrieves the Help context of a member of the ActiveX control.

Pascal

```
function GetHelpContext(Member: string; var HelpCtx: Integer; var HelpFile: string): Boolean;
```

C++

```
__fastcall Boolean GetHelpContext(UnicodeString Member, int HelpCtx, UnicodeString HelpFile);
```

Description

Use GetHelpContext to retrieve the Help context of a member of the ActiveX control.

Member is the name of a property, method, or event of the ActiveX control. To retrieve the help context of the ActiveX control itself, set Member to nil (Delphi) or NULL (C++).

HelpCtx is a variable in which the Help context identifier of Member is to be stored.

HelpFile is a variable in which the name of the Help file containing HelpCtx is to be stored.

The information about the ActiveX control is retrieved from its ITypeInfo interfaces. For more information, search the OLE.HLP file for ITypeInfo.

GetHelpContext returns false if Member is not found; otherwise, GetHelpContext returns true.

48.1.4.4.29 TOleControl.GetIDispatchProp Method

Reads a property of an ActiveX control. The property is expected to be of type IDispatch.

Pascal

```
function GetIDispatchProp(Index: Integer): IDispatch;
```

C++

```
__fastcall IDispatch GetIDispatchProp(int Index);
```

Description

Use GetIDispatchProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetIDispatchProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.30 TOleControl.GetIDsOfNames Method

Maps a single member and an optional set of argument names to a corresponding set of integer dispatch identifiers (dispIDs).

Pascal

```
function GetIDsOfNames(const IID: TGUID; Names: Pointer; NameCount: Integer; LocaleID:
Integer; DispIDs: Pointer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetIDsOfNames(const TGUID IID, void * Names, int NameCount, int
LocaleID, void * DispIDs);
```

Description

GetIDsOfNames implements the IDispatch interface GetIDsOfNames method. For components that support interfaces, GetIDsOfNames calls this method for the interface supported by the component, passing the specified parameters. The returned value can be used on subsequent calls to the Invoke method.

See Also

GetTypeInfo

GetTypeInfoCount

Invoke

48.1.4.4.31 TOleControl.GetIntegerProp Method

Reads a property of an ActiveX control. The property is expected to be of type `Integer`.

Pascal

```
function GetIntegerProp(Index: Integer): Integer;
```

C++

```
__fastcall int GetIntegerProp(int Index);
```

Description

Use GetIntegerProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetIntegerProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.32 TOleControl.GetIUnknownProp Method

Reads a property of an ActiveX control. The property is expected to be of type IUnknown.

Pascal

```
function GetIUnknownProp(Index: Integer): IUnknown;
```

C++

```
__fastcall IUnknown GetIUnknownProp(int Index);
```

Description

Use GetIUnknownProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetIUnknownProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.33 TOleControl.GetMoniker Method

Pascal

```
function GetMoniker(dwAssign: Longint; dwWhichMoniker: Longint; out mk: IMoniker): HResult;
stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetMoniker(Longint dwAssign, Longint dwWhichMoniker, IMoniker mk);
```

Description

This is GetMoniker, a member of class TOleControl.

48.1.4.4.34 TOleControl.GetObjectVerbs Method

Retrieves the names of the verbs implemented by the ActiveX control.

Pascal

```
procedure GetObjectVerbs(List: TStrings);
```

C++

```
__fastcall GetObjectVerbs(TStrings List);
```

Description

Use GetObjectVerbs to retrieve the names of the verbs implemented by the ActiveX control.

List is where the names of the verbs are to be stored.

48.1.4.4.35 TOleControl.GetOleBoolProp Method

Reads a property of an ActiveX control. The property is expected to be of type `TOleBool`.

Pascal

```
function GetOleBoolProp(Index: Integer): TOleBool;
```

C++

```
__fastcall TOleBool GetOleBoolProp(int Index);
```

Description

Use GetOleBoolProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetOleBoolProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.36 TOleControl.GetOleDateProp Method

Reads a property of an ActiveX control. The property is expected to be of type `TOleDate`.

Pascal

```
function GetOleDateProp(Index: Integer): TOleDate;
```

C++

```
__fastcall TOleDate GetOleDateProp(int Index);
```

Description

Use GetOleDateProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetOleDateProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.37 TOleControl.GetOleEnumProp Method

Reads a property of an ActiveX control. The property is expected to be of type `TOleEnum (see page 2705)`.

Pascal

```
function GetOleEnumProp(Index: Integer): TOleEnum;
```

C++

```
__fastcall TOleEnum GetOleEnumProp(int Index);
```

Description

Use GetOleEnumProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetOleEnumProp returns the value of the read property.

See Also

GetTOleEnumProp (see page 2664)

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.38 TOleControl.GetOleVariantProp Method

Reads a property of an ActiveX control. The property is expected to be of type `OleVariant`.

Pascal

```
function GetOleVariantProp(Index: Integer): OleVariant;
```

C++

```
__fastcall OleVariant GetOleVariantProp(int Index);
```

Description

Use GetOleVariantProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetOleVariantProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.39 TOleControl.GetPropDisplayString Method

Returns the display string for an interface for the ActiveX control.

Pascal

```
function GetPropDisplayString(DispID: Integer): string;
```

C++

```
__fastcall UnicodeString GetPropDisplayString(int DispID);
```

Description

GetPropDisplayString returns the display string for an interface for the ActiveX control.

DispID is the identifier for the interface. For more information, search the OLE.HLP file for DISPID.

For more information about GetPropDisplayString, search the latest Microsoft Developer Network CD for GetDisplayStrings.

See Also

GetPropDisplayStrings (see page 2661)

SetPropDisplayString (see page 2683)

48.1.4.4.40 TOleControl.GetPropDisplayStrings Method

Returns the display strings for an interface for the ActiveX control.

Pascal

```
procedure GetPropDisplayStrings(DispID: Integer; List: TStrings);
```

C++

```
__fastcall GetPropDisplayStrings(int DispID, TStrings List);
```

Description

List is where the strings are to be stored.

For more information about GetPropDisplayStrings, search the latest Microsoft Developer Network CD for GetDisplayStrings.

See Also

GetPropDisplayString (see page 2661)

48.1.4.4.41 TOleControl.GetProperty Method

Reads a property of an ActiveX control.

Pascal

```
procedure GetProperty(Index: Integer; var Value: TVarData);
```

C++

```
__fastcall GetProperty(int Index, TVarData Value);
```

Description

Use GetProperty to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. The `Value` output parameter is a TVarData, which will be filled with the read property value.

See Also

GetOleVariantProp (see page 2660)

GetVariantProp (see page 2666)

48.1.4.4.42 TOleControl.GetShortIntProp Method

Reads a property of an ActiveX control. The property is expected to be of type `ShortInt`.

Pascal

```
function GetShortIntProp(Index: Integer): ShortInt;
```

C++

```
__fastcall ShortInt GetShortIntProp(int Index);
```

Description

Use GetShortIntProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetShortIntProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.43 TOleControl.GetSingleProp Method

Reads a property of an ActiveX control. The property is expected to be of type `Single`.

Pascal

```
function GetSingleProp(Index: Integer): Single;
```

C++

```
__fastcall Single GetSingleProp(int Index);
```

Description

Use GetSingleProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetSingleProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.44 TOleControl.GetSmallintProp Method

Reads a property of an ActiveX control. The property is expected to be of type `Smallint`.

Pascal

```
function GetSmallintProp(Index: Integer): Smallint;
```

C++

```
__fastcall Smallint GetSmallintProp(int Index);
```

Description

Use GetSmallintProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetSmallintProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.45 TOleControl.GetStringProp Method

Reads a property of an ActiveX control. The property is expected to be of type `String`.

Pascal

```
function GetStringProp(Index: Integer): string;
```

C++

```
__fastcall UnicodeString GetStringProp(int Index);
```

Description

Use GetStringProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetStringProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.46 TOleControl.GetTColorProp Method

Reads a property of an ActiveX control. The property is expected to be of type TColor.

Pascal

```
function GetTColorProp(Index: Integer): TColor;
```

C++

```
__fastcall TColor GetTColorProp(int Index);
```

Description

Use GetTColorProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetTColorProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetColorProp (see page 2655)

GetVariantProp (see page 2666)

48.1.4.4.47 TOleControl.GetTDateTimeProp Method

Reads a property of an ActiveX control. The property is expected to be of type TDateTime.

Pascal

```
function GetTDateTimeProp(Index: Integer): TDateTime;
```

C++

```
__fastcall TDateTime GetTDateTimeProp(int Index);
```

Description

Use GetTDateTimeProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetTDateTimeProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.48 TOleControl.GetTFontProp Method

Reads a property of an ActiveX control. The property is expected to be of type TFont.

Pascal

```
function GetTFontProp(Index: Integer): TFont;
```

C++

```
__fastcall TFont GetTFontProp(int Index);
```

Description

Use GetTFontProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetTFontProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.49 TOleControl.GetTOleEnumProp Method

Reads a property of an ActiveX control. The property is expected to be of type `TOleEnum (see page 2705)`.

Pascal

```
function GetTOleEnumProp(Index: Integer): TOleEnum;
```

C++

```
__fastcall TOleEnum GetTOleEnumProp(int Index);
```

Description

Use GetTOleEnumProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetTOleEnumProp returns the value of the read property.

See Also

GetOleEnumProp (see page 2660)

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.50 TOleControl.GetTPictureProp Method

Reads a property of an ActiveX control. The property is expected to be of type TPicture.

Pascal

```
function GetTPictureProp(Index: Integer): TPicture;
```

C++

```
__fastcall TPicture GetTPictureProp(int Index);
```

Description

Use GetTPictureProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetTPictureProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.51 TOleControl.GetTypeInfo Method

Retrieves the type information for an object.

Pascal

```
function GetTypeInfo(Index: Integer; LocaleID: Integer; out TypeInfo): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetTypeInfo(int Index, int LocaleID,  TypeInfo);
```

Description

GetTypeInfo implements the IDispatch interface GetTypeInfo method. For components that support interfaces, GetTypeInfo calls the GetTypeInfo method for the interface supported by the component, passing it the specified parameters. Use the returned value to get the type information for an interface implemented by the component.

See Also

GetIDsOfNames

GetTypeInfoCount

Invoke

48.1.4.4.52 TOleControl.GetTypeInfoCount Method

Returns the number of type information interfaces that an object provides (either 0 or 1).

Pascal

```
function GetTypeInfoCount(out Count: Integer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetTypeInfoCount(int Count);
```

Description

GetTypeInfoCount implements the IDispatch interface GetTypeInfoCount method. For components that support interfaces, GetTypeInfoCount calls this method for the interface supported by the component. The `Count` parameter points to a location that receives the number of type information interfaces provided by the object. If the object provides type information, this number is 1; otherwise the number is 0.

See Also

GetIDsOfNames

GetTypeInfo

Invoke

48.1.4.4.53 TOleControl.GetVariantProp Method

Reads a property of an ActiveX control.

Pascal

```
function GetVariantProp(Index: Integer): Variant;
```

C++

```
__fastcall Variant GetVariantProp(int Index);
```

Description

Use GetVariantProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. The `Value` output parameter is a `Variant` that will be filled with the read property value.

See Also

GetOleVariantProp (see page 2660)

GetProperty (see page 2661)

48.1.4.4.54 TOleControl.GetWideStringProp Method

Reads a property of an ActiveX control. The property is expected to be of type `WideString`.

Pascal

```
function GetWideStringProp(Index: Integer): WideString;
```

C++

```
__fastcall BSTR GetWideStringProp(int Index);
```

Description

Use GetWideStringProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetWideStringProp returns the value of the read property.

See Also

GetStringProp (see page 2663)

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.55 TOleControl.GetWindowContext Method

Pascal

```
function GetWindowContext(out frame: IOleInPlaceFrame; out doc: IOleInPlaceUIWindow; out
rcPosRect: TRect; out rcClipRect: TRect; out frameInfo: TOleInPlaceFrameInfo): HResult;
stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetWindowContext(IOleInPlaceFrame frame, IOleInPlaceUIWindow doc,
TRect rcPosRect, TRect rcClipRect, TOleInPlaceFrameInfo frameInfo);
```

Description

This is GetWindowContext, a member of class TOleControl.

48.1.4.4.56 TOleControl.GetWordBoolProp Method

Reads a property of an ActiveX control. The property is expected to be of type `WordBool`.

Pascal

```
function GetWordBoolProp(Index: Integer): WordBool;
```

C++

```
__fastcall WordBool GetWordBoolProp(int Index);
```

Description

Use GetWordBoolProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetWordBoolProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.57 TOleControl.GetWordProp Method

Reads a property of an ActiveX control. The property is expected to be of type `Word`.

Pascal

```
function GetWordProp(Index: Integer): Word;
```

C++

```
__fastcall Word GetWordProp(int Index);
```

Description

Use GetWordProp to read a property of an ActiveX control. The `Index` parameter specifies the **dispid** of the property being read. GetWordProp returns the value of the read property.

See Also

GetProperty (see page 2661)

GetVariantProp (see page 2666)

48.1.4.4.58 TOleControl.InitControlData Method

Pascal

```
procedure InitControlData; virtual; abstract;
```

C++

```
virtual __fastcall InitControlData() = 0;
```

Description

This is InitControlData, a member of class TOleControl.

48.1.4.4.59 TOleControl.InitControlInterface Method

Pascal

```
procedure InitControlInterface(const Obj: IUnknown); virtual;
```

C++

```
virtual __fastcall InitControlInterface(const IUnknown Obj);
```

Description

This is InitControlInterface, a member of class TOleControl.

48.1.4.4.60 TOleControl.InsertMenus Method

Pascal

```
function InsertMenus(hmenuShared: HMenu; var menuWidths: TOleMenuGroupWidths): HResult;
stdcall;
```

C++

```
__fastcall __stdcall HRESULT InsertMenus(HMenu hmenuShared, TOleMenuGroupWidths menuWidths);
```

Description

This is InsertMenus, a member of class TOleControl.

48.1.4.4.61 TOleControl.Invoke Method

Provides access to Automation properties and methods when the component wraps an Automation object.

Pascal

```
function Invoke(DispID: Integer; const IID: TGUID; LocaleID: Integer; Flags: Word; var Params;
VarResult: Pointer; ExcepInfo: Pointer; ArgErr: Pointer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT Invoke(int DispID, const TGUID IID, int LocaleID, Word Flags,
Params, void * VarResult, void * ExcepInfo, void * ArgErr);
```

Description

Invoke is the standard mechanism for accessing the exposed properties and methods of an Automation object. For components that wrap the IDispatch interface of an Automation object, Invoke calls the Invoke method for the interface supported by the component, passing it the parameters specified by the function.

See Also

GetIDsOfNames

GetTypeInfo

GetTypeInfoCount

ComObject

48.1.4.4.62 TOleControl.InvokeEvent Method

Pascal

```
procedure InvokeEvent(DispID: TDispID; var Params: TDispParams); virtual;
```

C++

```
virtual __fastcall InvokeEvent(TDispID DispID, TDispParams Params);
```

Description

This is InvokeEvent, a member of class TOleControl.

48.1.4.4.63 TOleControl.InvokeMethod Method

Pascal

```
procedure InvokeMethod(const DispInfo; Result: Pointer);
```

C++

```
__fastcall InvokeMethod(const  DispInfo, void * Result);
```

Description

This is InvokeMethod, a member of class TOleControl.

48.1.4.4.64 TOleControl.IsCustomProperty Method

Indicates whether an interface for the ActiveX control is a custom property.

Pascal

```
function IsCustomProperty(DispID: Integer): Boolean;
```

C++

```
__fastcall Boolean IsCustomProperty(int DispID);
```

Description

Use IsCustomProperty to determine whether an interface for the ActiveX control is a custom property.

DispID is the identifier for the interface. For more information, search the OLE.HLP file for DISPID.

IsCustomProperty returns true if an interface is a custom property; otherwise, IsCustomProperty returns false.

48.1.4.4.65 TOleControl.IsPropPageProperty Method

Indicates whether an interface for the OCX control is a property page property.

Pascal

```
function IsPropPageProperty(DispID: Integer): Boolean;
```

C++

```
__fastcall Boolean IsPropPageProperty(int DispID);
```

Description

Use IsPropPageProperty to determine whether an interface for the OCX control is a property page property.

DispID is the identifier for the interface. For more information, search the OLE.HLP file for DISPID.

IsPropPageProperty returns true if an interface is a property page property; otherwise, IsPropPageProperty returns false.

48.1.4.4.66 TOleControl.LockInPlaceActive Method

Pascal

```
function LockInPlaceActive(fLock: BOOL): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT LockInPlaceActive(BOOL fLock);
```

Description

This is LockInPlaceActive, a member of class TOleControl.

48.1.4.4.67 TOleControl.OleControlSite_TranslateAccelerator Method

Pascal

```
function OleControlSite_TranslateAccelerator(msg: PMsg; grfModifiers: Longint): HResult;
stdcall;
```

C++

```
__fastcall __stdcall HRESULT OleControlSite_TranslateAccelerator(PMsg msg, Longint
grfModifiers);
```

Description

This is OleControlSite_TranslateAccelerator, a member of class TOleControl.

48.1.4.4.68 TOleControl.OleInPlaceFrame_GetWindow Method

Pascal

```
function OleInPlaceFrame_GetWindow(out wnd: HWnd): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT OleInPlaceFrame_GetWindow(HWnd wnd);
```

Description

This is OleInPlaceFrame_GetWindow, a member of class TOleControl.

48.1.4.4.69 TOleControl.OleInPlaceFrame_TranslateAccelerator Method

Pascal

```
function OleInPlaceFrame_TranslateAccelerator(var msg: TMsg; wID: Word): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT OleInPlaceFrame_TranslateAccelerator(TMsg msg, Word wID);
```

Description

This is OleInPlaceFrame_TranslateAccelerator, a member of class TOleControl.

48.1.4.4.70 TOleControl.OleInPlaceSite_GetWindow Method

Pascal

```
function OleInPlaceSite_GetWindow(out wnd: HWnd): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT OleInPlaceSite_GetWindow(HWnd wnd);
```

Description

This is OleInPlaceSite_GetWindow, a member of class TOleControl.

48.1.4.4.71 TOleControl.OnChanged Method

Pascal

```
function OnChanged(dispid: TDispID): HResult; virtual; stdcall;
```

C++

```
virtual __fastcall __stdcall HRESULT OnChanged(TDispID dispid);
```

Description

This is OnChanged, a member of class TOleControl.

48.1.4.4.72 TOleControl.OnControlInfoChanged Method

Pascal

```
function OnControlInfoChanged: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT OnControlInfoChanged();
```

Description

This is OnControlInfoChanged, a member of class TOleControl.

48.1.4.4.73 TOleControl.OnFocus Method

Pascal

```
function OnFocus(fGotFocus: BOOL): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT OnFocus(BOOL fGotFocus);
```

Description

This is OnFocus, a member of class TOleControl.

48.1.4.4.74 TOleControl.OnInPlaceActivate Method

Pascal

```
function OnInPlaceActivate: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT OnInPlaceActivate();
```

Description

This is OnInPlaceActivate, a member of class TOleControl.

48.1.4.4.75 TOleControl.OnInPlaceDeactivate Method

Pascal

```
function OnInPlaceDeactivate: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT OnInPlaceDeactivate();
```

Description

This is OnInPlaceDeactivate, a member of class TOleControl.

48.1.4.4.76 TOleControl.OnPosRectChange Method

Pascal

```
function OnPosRectChange(const rcPosRect: TRect): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT OnPosRectChange(const TRect rcPosRect);
```

Description

This is OnPosRectChange, a member of class TOleControl.

48.1.4.4.77 TOleControl.OnRequestEdit Method

Pascal

```
function OnRequestEdit(dispid: TDispID): HResult; virtual; stdcall;
```

C++

```
virtual __fastcall __stdcall HRESULT OnRequestEdit(TDispID dispid);
```

Description

This is OnRequestEdit, a member of class TOleControl.

48.1.4.4.78 TOleControl.OnShowWindow Method

Pascal

```
function OnShowWindow(fShow: BOOL): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT OnShowWindow(BOOL fShow);
```

Description

This is OnShowWindow, a member of class TOleControl.

48.1.4.4.79 TOleControl.OnUIActivate Method

Pascal

```
function OnUIActivate: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT OnUIActivate();
```

Description

This is OnUIActivate, a member of class TOleControl.

48.1.4.4.80 TOleControl.OnUIDeactivate Method

Pascal

```
function OnUIDeactivate(fUndoable: BOOL): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT OnUIDeactivate(BOOL fUndoable);
```

Description

This is OnUIDeactivate, a member of class TOleControl.

48.1.4.4.81 TOleControl.PaletteChanged Method

Responds to changes in the system's palette by realizing the control's palette and the palette for each child control.

Pascal

```
function PaletteChanged(Foreground: Boolean): Boolean; override;
```

C++

```
virtual __fastcall Boolean PaletteChanged(Boolean Foreground);
```

Description

PaletteChanged is called automatically when the Windows system palette changes. TWinControl responds to these notifications by trying to realize the windowed control's palette, if any, into the current device context, and then passing on the PaletteChanged notification to each of the child controls in turn. If the change in the system palette causes the actual value of the control's palette to change, PaletteChanged will invalidate the control so that it can repaint with the new palette.

Windows paints the active window with a foreground palette, while other windows are forced to use background palettes. Background palettes are approximate matches to the colors specified by the control's logical palette, given the limitations imposed by implementing the foreground palette. Windows only allows a single foreground palette.

PaletteChanged allows the control to obtain a new realization of its palette when the system palette changes. When Foreground is true, the form has been activated and the control is specifying the new foreground palette. When Foreground is false, another application or form has changed the foreground palette, and controls that are sensitive to the available palette should realize new background palettes to best match their logical palettes.

If the control does not have a logical palette to be realized into the current device context, GetPalette will return a handle of 0, and PaletteChanged will not try to realize a palette for the windowed control. Instead, it simply passes the notification on to the child controls. If the windowed control or any of its child controls realize a palette into the current device context in response to

PaletteChanged, this method returns true.

Override PaletteChanged to change the way in which controls are given an opportunity to realize a new palette when the Windows system palette changes. For example, override PaletteChanged to change the order in which child controls realize their palettes, giving a particular control the foreground palette when the form is activated, or to pass palette change notification to custom controls implemented outside of the VCL, such as in DLLs.

Override GetPalette instead to affect whether the windowed control actually has a palette to realize, or to change the value of the control's logical palette.

Note: PaletteChanged is called only when the run-time video mode requires palette support, such as for 256 color mode, but not 16 million color mode.

See Also

GetDeviceContext

GetPalette

48.1.4.4.82 TOleControl.PictureChanged Method

Pascal

```
procedure PictureChanged(Sender: TObject);
```

C++

```
__fastcall PictureChanged(TObject * Sender);
```

Description

This is PictureChanged, a member of class TOleControl.

48.1.4.4.83 TOleControl.PostMessageFilter Method

Pascal

```
function PostMessageFilter(wnd: HWnd; msg: Integer; wp: Integer; lp: Integer; out res:
Integer; Cookie: Longint): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT PostMessageFilter(HWnd wnd, int msg, int wp, int lp, int res,
Longint Cookie);
```

Description

This is PostMessageFilter, a member of class TOleControl.

48.1.4.4.84 TOleControl.PreMessageFilter Method

Pascal

```
function PreMessageFilter(wnd: HWnd; msg: Integer; wp: Integer; lp: Integer; out res: Integer;
out Cookie: Longint): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT PreMessageFilter(HWnd wnd, int msg, int wp, int lp, int res,
Longint Cookie);
```

Description

This is PreMessageFilter, a member of class TOleControl.

48.1.4.4.85 TOleControl.QueryInterface Method

Returns a reference to a specified interface if the object supports that interface.

Pascal

```
function QueryInterface(const IID: TGUID; out Obj): HResult; override;
```

C++

```
virtual __fastcall HRESULT QueryInterface(const TGUID IID,  Obj);
```

Description

QueryInterface checks whether the component supports the interface specified by IID, and if so, returns a reference to that interface as the Obj parameter. If the component does not support the interface, the Obj parameter returns nil (Delphi) or NULL (C++).

Note: For components that act as COM object wrapper, QueryInterface calls the QueryInterface method of the internal COM object.

See Also

_AddRef

_Release

QueryInterface

48.1.4.4.86 TOleControl.QueryService Method

Pascal

```
function QueryService(const rsid: TGuid; const iid: TGuid; out obj): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT QueryService(const TGuid rsid, const TGuid iid,  obj);
```

Description

This is QueryService, a member of class TOleControl.

48.1.4.4.87 TOleControl.RemoveMenus Method

Pascal

```
function RemoveMenus(hmenuShared: HMenu): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT RemoveMenus(HMenu hmenuShared);
```

Description

This is RemoveMenus, a member of class TOleControl.

48.1.4.4.88 TOleControl.RequestBorderSpace Method

Pascal

```
function RequestBorderSpace(const borderwidths: TRect): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT RequestBorderSpace(const TRect borderwidths);
```

Description

This is RequestBorderSpace, a member of class TOleControl.

48.1.4.4.89 TOleControl.RequestNewObjectLayout Method

Pascal

```
function RequestNewObjectLayout: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT RequestNewObjectLayout();
```

Description

This is RequestNewObjectLayout, a member of class TOleControl.

48.1.4.4.90 TOleControl.SaveObject Method

Saves the object to persistent storage.

Pascal

```
function SaveObject: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT SaveObject();
```

Description

SaveObject is called by an enclosed ActiveX control whenever saving it is required. The call blocks until the operation completes. This guarantees that the object has been stored before the call returns.

See Also

ShowObject (see page 2689)

GetContainer (see page 2655)

48.1.4.4.91 TOleControl.Scroll Method

Scrolls the view by the indicated amount of pixels.

Pascal

```
function Scroll(scrollExtent: TPoint): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT Scroll(TPoint scrollExtent);
```

Description

Call Scroll to scroll the view of the ActiveX control by the indicated amount of pixels.

See Also

SetBounds (see page 2677)

GetBorder (see page 2654)

48.1.4.4.92 TOleControl.SetActiveObject Method

Pascal

```
function SetActiveObject(const activeObject: IOleInPlaceActiveObject; pszObjName: POleStr):
HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT SetActiveObject(const IOleInPlaceActiveObject activeObject,
POleStr pszObjName);
```

Description

This is SetActiveObject, a member of class TOleControl.

48.1.4.4.93 TOleControl.SetBorderSpace Method

Allocates border space for the ActiveX control.

Pascal

```
function SetBorderSpace(pborderwidths: PRect): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT SetBorderSpace(PRect pborderwidths);
```

Description

Call SetBorderSpace to allocate border space for the ActiveX control. You should perform a call to RequestBorderSpace (see page 2675) prior to a call to SetBorderSpace.

See Also

ShowPropertyFrame (see page 2689)

SetBounds (see page 2677)

RequestBorderSpace (see page 2675)

48.1.4.4.94 TOleControl.SetBounds Method

Sets the boundaries of the ActiveX control.

Pascal

```
procedure SetBounds(ALeft: Integer; ATop: Integer; AWidth: Integer; AHeight: Integer);
override;
```

C++

```
virtual __fastcall SetBounds(int ALeft, int ATop, int AWidth, int AHeight);
```

Description

Use SetBounds to set the boundaries of the ActiveX control.

ALeft, ATop, AWidth, and AHeight are the x-coordinate of the upper-left corner, the y-coordinate of the upper-left corner, the width and the height of the ActiveX control.

48.1.4.4.95 TOleControl.SetByteProp Method

Sets a property of an ActiveX control to the specified value of type `Byte`.

Pascal

```
procedure SetByteProp(Index: Integer; Value: Byte);
```

C++

```
__fastcall SetByteProp(int Index, Byte Value);
```

Description

Use SetByteProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.96 TOleControl.SetColorProp Method

Sets a property of an ActiveX control to the specified color.

Pascal

```
procedure SetColorProp(Index: Integer; Value: TColor);
```

C++

```
__fastcall SetColorProp(int Index, TColor Value);
```

Description

Use SetColorProp to set a property of an ActiveX control to the specified color. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set color.

See Also

_SetTColorProp (see page 2648)

SetTColorProp (see page 2685)

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.97 TOleControl.SetCompProp Method

Sets a property of an ActiveX control to the specified value of type `Comp`.

Pascal

```
procedure SetCompProp(Index: Integer; const Value: Comp);
```

C++

```
__fastcall SetCompProp(int Index, const Comp Value);
```

Description

Use SetCompProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.98 TOleControl.SetCurrencyProp Method

Sets a property of an ActiveX control to the specified value of type `Currency`.

Pascal

```
procedure SetCurrencyProp(Index: Integer; const Value: Currency);
```

C++

```
__fastcall SetCurrencyProp(int Index, const Currency Value);
```

Description

Use SetCurrencyProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.99 TOleControl.SetDoubleProp Method

Sets a property of an ActiveX control to the specified value of type `Double`.

Pascal

```
procedure SetDoubleProp(Index: Integer; const Value: Double);
```

C++

```
__fastcall SetDoubleProp(int Index, const Double Value);
```

Description

Use SetDoubleProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.100 TOleControl.SetIDispatchProp Method

Sets a property of an ActiveX control to the specified value of type IDispatch.

Pascal

```
procedure SetIDispatchProp(Index: Integer; const Value: IDispatch);
```

C++

```
__fastcall SetIDispatchProp(int Index, const IDispatch Value);
```

Description

Use SetIDispatchProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.101 TOleControl.SetIntegerProp Method

Sets a property of an ActiveX control to the specified value of type `Integer`.

Pascal

```
procedure SetIntegerProp(Index: Integer; Value: Integer);
```

C++

```
__fastcall SetIntegerProp(int Index, int Value);
```

Description

Use SetIntegerProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.102 TOleControl.SetIUnknownProp Method

Sets a property of an ActiveX control to the specified value of type IUnknown.

Pascal

```
procedure SetIUnknownProp(Index: Integer; const Value: IUnknown);
```

C++

```
__fastcall SetIUnknownProp(int Index, const IUnknown Value);
```

Description

Use SetIUnknownProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.103 TOleControl.SetMenu Method

Attaches a menu to the in-place activated ActiveX object.

Pascal

```
function SetMenu(hmenuShared: HMenu; holemenu: HMenu; hwndActiveObject: HWnd): HResult;
stdcall;
```

C++

```
__fastcall __stdcall HRESULT SetMenu(HMenu hmenuShared, HMenu holemenu, HWnd hwndActiveObject);
```

Description

Call SetMenu to attach a menu to the ActiveX control, which is in-place activated. To remove the menu, call SetMenu with `hmenuShared` set to 0. Removing the menu is mandatory when the ActiveX object is to be deactivated.

See Also

ShowPropertyFrame (see page 2689)

SetBounds (see page 2677)

GetBorder (see page 2654)

48.1.4.4.104 TOleControl.SetName Method

Sets the name of the control.

Pascal

```
procedure SetName(const Value: TComponentName); override;
```

C++

```
virtual __fastcall SetName(const TComponentName Value);
```

Description

SetName is the protected write implementation of the Name property. TControl updates the Text or Caption property, if appropriate, after changing the value of Name.

Override SetName to execute additional code when the value of the Name property changes.

See Also

Name

Caption

Text

48.1.4.4.105 TOleControl.SetOleBoolProp Method

Sets a property of an ActiveX control to the specified value of type `TOleBool`.

Pascal

```
procedure SetOleBoolProp(Index: Integer; Value: TOleBool);
```

C++

```
__fastcall SetOleBoolProp(int Index, TOleBool Value);
```

Description

Use SetOleBoolProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.106 TOleControl.SetOleDateProp Method

Sets a property of an ActiveX control to the specified value of type `TOleDate`.

Pascal

```
procedure SetOleDateProp(Index: Integer; const Value: TOleDate);
```

C++

```
__fastcall SetOleDateProp(int Index, const TOleDate Value);
```

Description

Use SetOleDateProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.107 TOleControl.SetOleEnumProp Method

Sets a property of an ActiveX control to the specified value of type TOleEnum (see page 2705).

Pascal

```
procedure SetOleEnumProp(Index: Integer; Value: TOleEnum);
```

C++

```
__fastcall SetOleEnumProp(int Index, TOleEnum Value);
```

Description

Use SetOleEnumProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.108 TOleControl.SetOleVariantProp Method

Sets a property of an ActiveX control to the specified value of type `OleVariant`.

Pascal

```
procedure SetOleVariantProp(Index: Integer; const Value: OleVariant);
```

C++

```
__fastcall SetOleVariantProp(int Index, const OleVariant Value);
```

Description

Use SetOleVariantProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.109 TOleControl.SetParent Method

Sets the parent of the control.

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

SetParent is the protected implementation of the Parent property. Override SetParent to execute additional code when the value of the Parent property changes.

The AParent parameter specifies the new parent of the control.

If the control already has a parent, SetParent removes the control from that parent's list of controls by calling the parent's RemoveControl method. If AParent is not nil (Delphi) or NULL (C++), SetParent calls its InsertControl method to add the control to its list of controls.

The TControl destructor calls SetParent(nil (Delphi) or NULL (C++)) to remove the control from its parent's control list before destroying the component.

See Also

Destroy

InsertControl

Parent

RemoveControl

48.1.4.4.110 TOleControl.SetPropDisplayString Method

Sets the display string for an interface for the ActiveX control.

Pascal

```
procedure SetPropDisplayString(DispID: Integer; const Value: string);
```

C++

```
__fastcall SetPropDisplayString(int DispID, const UnicodeString Value);
```

Description

Use SetPropDisplayString to set the display string for an interface for the ActiveX control.

DispID is the identifier for the interface. For more information, search the OLE.HLP file for DISPID.

Value is the new display string for the interface.

See Also

GetPropDisplayStrings (see page 2661)

48.1.4.4.111 TOleControl.SetProperty Method

Sets a property of an ActiveX control to the specified value.

Pascal

```
procedure SetProperty(Index: Integer; const Value: TVarData);
```

C++

```
__fastcall SetProperty(int Index, const TVarData Value);
```

Description

Use SetProperty to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetOleVariantProp (see page 2682)

SetVariantProp (see page 2687)

48.1.4.4.112 TOleControl.SetShortIntProp Method

Sets a property of an ActiveX control to the specified value of type `ShortInt`.

Pascal

```
procedure SetShortIntProp(Index: Integer; Value: Shortint);
```

C++

```
__fastcall SetShortIntProp(int Index, Shortint Value);
```

Description

Use SetShortIntProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.113 TOleControl.SetSingleProp Method

Sets a property of an ActiveX control to the specified value of type `Single`.

Pascal

```
procedure SetSingleProp(Index: Integer; const Value: Single);
```

C++

```
__fastcall SetSingleProp(int Index, const Single Value);
```

Description

Use SetSingleProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.114 TOleControl.SetSmallintProp Method

Sets a property of an ActiveX control to the specified value of type `Smallint`.

Pascal

```
procedure SetSmallintProp(Index: Integer; Value: Smallint);
```

C++

```
__fastcall SetSmallintProp(int Index, Smallint Value);
```

Description

Use SetSmallintProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.115 TOleControl.SetStatusText Method

Pascal

```
function SetStatusText(pszStatusText: POleStr): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT SetStatusText(POleStr pszStatusText);
```

Description

This is SetStatusText, a member of class TOleControl.

48.1.4.4.116 TOleControl.SetStringProp Method

Sets a property of an ActiveX control to the specified value of type `String`.

Pascal

```
procedure SetStringProp(Index: Integer; const Value: string);
```

C++

```
__fastcall SetStringProp(int Index, const UnicodeString Value);
```

Description

Use SetStringProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.117 TOleControl.SetTColorProp Method

Sets a property of an ActiveX control to the specified color.

Pascal

```
procedure SetTColorProp(Index: Integer; Value: TColor);
```

C++

```
__fastcall SetTColorProp(int Index, TColor Value);
```

Description

Use SetTColorProp to set a property of an ActiveX control to the specified color. The `Index` parameter specifies the

dispid of the property being set. The `Value` parameter contains the set color.

See Also

_SetTColorProp (see page 2648)

_SetColorProp (see page 2647)

SetColorProp (see page 2678)

48.1.4.4.118 TOleControl.SetTDateTimeProp Method

Sets a property of an ActiveX control to the specified value of type TDateTime.

Pascal

```
procedure SetTDateTimeProp(Index: Integer; const Value: TDateTime);
```

C++

```
__fastcall SetTDateTimeProp(int Index, const TDateTime Value);
```

Description

Use SetTDateTimeProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.119 TOleControl.SetTFontProp Method

Sets a property of an ActiveX control to the specified font.

Pascal

```
procedure SetTFontProp(Index: Integer; Value: TFont);
```

C++

```
__fastcall SetTFontProp(int Index, TFont Value);
```

Description

Use SetTFontProp to set a property of an ActiveX control to the specified font. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set font.

See Also

_SetTFontProp (see page 2648)

SetTColorProp (see page 2685)

SetTPictureProp (see page 2687)

SetTOleEnumProp (see page 2686)

48.1.4.4.120 TOleControl.SetTOleEnumProp Method

Sets a property of an ActiveX control to the specified enumeration value.

Pascal

```
procedure SetTOleEnumProp(Index: Integer; Value: TOleEnum);
```

C++

```
__fastcall SetTOleEnumProp(int Index, TOleEnum Value);
```

Description

Use SetTOleEnumProp to set a property of an ActiveX control to the specified enumeration value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set enumeration value.

See Also

_SetTOleEnumProp (see page 2649)

SetTFontProp (see page 2686)

SetTColorProp (see page 2685)

SetTPictureProp (see page 2687)

48.1.4.4.121 TOleControl.SetTPictureProp Method

Sets a property of an ActiveX control to the specified picture object.

Pascal

```
procedure SetTPictureProp(Index: Integer; Value: TPicture);
```

C++

```
__fastcall SetTPictureProp(int Index, TPicture Value);
```

Description

Use SetTPictureProp to set a property of an ActiveX control to the specified picture object. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set picture object.

See Also

_SetTPictureProp (see page 2649)

SetTColorProp (see page 2685)

SetTFontProp (see page 2686)

SetTOleEnumProp (see page 2686)

48.1.4.4.122 TOleControl.SetVariantProp Method

Sets a property of an ActiveX control to the specified value.

Pascal

```
procedure SetVariantProp(Index: Integer; const Value: Variant);
```

C++

```
__fastcall SetVariantProp(int Index, const Variant Value);
```

Description

Use SetVariantProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

48.1.4.4.123 TOleControl.SetWideStringProp Method

Sets a property of an ActiveX control to the specified value of type `WideString`.

Pascal

```
procedure SetWideStringProp(Index: Integer; const Value: WideString);
```

C++

```
__fastcall SetWideStringProp(int Index, const BSTR Value);
```

Description

Use SetWideStringProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.124 TOleControl.SetWordBoolProp Method

Sets a property of an ActiveX control to the specified value of type `WordBool`.

Pascal

```
procedure SetWordBoolProp(Index: Integer; Value: WordBool);
```

C++

```
__fastcall SetWordBoolProp(int Index, WordBool Value);
```

Description

Use SetWordBoolProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.125 TOleControl.SetWordProp Method

Sets a property of an ActiveX control to the specified value of type `Word`.

Pascal

```
procedure SetWordProp(Index: Integer; Value: Word);
```

C++

```
__fastcall SetWordProp(int Index, Word Value);
```

Description

Use SetWordProp to set a property of an ActiveX control to the specified value. The `Index` parameter specifies the **dispid** of the property being set. The `Value` parameter contains the set value.

See Also

SetProperty (see page 2683)

SetVariantProp (see page 2687)

48.1.4.4.126 TOleControl.ShowAboutBox Method

Notifies the ActiveX control to display its About box.

Pascal

```
procedure ShowAboutBox;
```

C++

```
__fastcall ShowAboutBox();
```

Description

Use ShowAboutBox to notify the ActiveX control to display its About box.

48.1.4.4.127 TOleControl.ShowObject Method

Makes the enclosed object visible.

Pascal

```
function ShowObject: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT ShowObject();
```

Description

Call ShowObject to make the enclosed ActiveX object visible. The enclosed ActiveX control may not be visible directly after the call. The call to ShowObject does not guarantee that the object becomes visible.

See Also

ShowPropertyFrame (see page 2689)

SetBounds (see page 2677)

GetBorder (see page 2654)

48.1.4.4.128 TOleControl.ShowPropertyFrame Method

Specifies whether the TOleControl (see page 2638) shows a property sheet.

Pascal

```
function ShowPropertyFrame: HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT ShowPropertyFrame();
```

Description

Call ShowPropertyFrame to check whether this ActiveX control shows a property sheet for the contained controls. This method always returns E_NOTIMPL, which means that this feature is not implemented in TOleControl (see page 2638).

See Also

ShowObject (see page 2689)

SetBounds (see page 2677)

GetBorder (see page 2654)

48.1.4.4.129 TOleControl.StandardEvent Method

Pascal

```
procedure StandardEvent(DispID: TDispID; var Params: TDispParams); virtual;
```

C++

```
virtual __fastcall StandardEvent(TDispID DispID, TDispParams Params);
```

Description

This is StandardEvent, a member of class TOleControl.

48.1.4.4.130 TOleControl.SuppressException Method

SuppressException suppresses OLE exceptions at design time.

Pascal

```
function SuppressException(E: Exception): Boolean; virtual;
```

C++

```
virtual __fastcall Boolean SuppressException(Exception E);
```

Description

SuppressException is called whenever an exception is thrown by this TOleControl (see page 2638). The implementation provided in TOleControl (see page 2638) suppresses all OLE exceptions thrown by the control at design time. To add custom logic to error suppressing, override SuppressException in descendant classes.

See Also

EOleSysError (see page 2499)

48.1.4.4.131 TOleControl.TransformCoords Method

Translates a set of coordinates.

Pascal

```
function TransformCoords(var ptlHimetric: TPoint; var ptfContainer: TPointF; flags: Longint):
HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT TransformCoords(TPoint ptlHimetric, TPointF ptfContainer, Longint
flags);
```

Description

Use TransformCoords to transform the OLE-style set of coordinates in the `ptlHimetric` record to a control-style set given by the `ptfContainer` record.

See Also

SetBounds (see page 2677)

GetBorder (see page 2654)

48.1.4.4.132 TOleControl.WndProc Method

Provides specific message responses for the control.

Pascal

```
procedure WndProc(var Message: TMessage); override;
```

C++

```
virtual __fastcall WndProc(TMessage Message);
```

Description

Override WndProc to change the initial Windows message handler for the control. The WindowProc property is initialized to point to the WndProc method.

WndProc for TWinControl overrides the inherited method to define responses for focus, mouse, and keyboard messages. It sends all others to its inherited WndProc.

When overriding WndProc to provide specialized responses to messages, call the inherited WndProc at the end to dispatch any unhandled messages.

C++ Examples:

```
/*
This example shows how to use the WndProc method and the
WindowProc property to subclass a custom control's window
procedure.  This example subclasses the window procedure of
a TListBox descendant to respond to a user-defined message
called WM_STYLEMESSAGE. The subclassed window procedure can
be turned on or off by pressing a radio button.
*/
class TMyListBoxDescendant : public TListBox
{
__published:    // IDE-managed Components
    void __fastcall SubClassWndProc(Messages::TMessage &Message);
    void __fastcall ToggleSubClass(bool On);
    void __fastcall OnDrawItemProc(
      TWinControl *Control, int Index, const TRect &Rect,
      TOwnerDrawState State);
private:    // User declarations
public:     // User declarations
    __fastcall TMyListBoxDescendant(TComponent* Owner);
};

TForm1 *Form1;
TMyListBoxDescendant *MyListBoxDescendant1;
Graphics::TBitmap *bitmap0;

const WM_STYLEMESSAGE = WM_USER + 2000;

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

__fastcall TMyListBoxDescendant::TMyListBoxDescendant(TComponent* Owner)
    : TListBox(Owner)
{
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
```

```
    PostMessage(
      MyListBoxDescendant1->Handle,
      WM_STYLEMESSAGE,
      Integer(lbOwnerDrawFixed),
      0);
}

void __fastcall TForm1::Button2Click(TObject *Sender)
{
  PostMessage(
    MyListBoxDescendant1->Handle,
    WM_STYLEMESSAGE,
    Integer(lbStandard),
    0);
}

void __fastcall TForm1::SubClassRadioGroup1Click(TObject *Sender)
{
  MyListBoxDescendant1->ToggleSubClass(SubClassRadioGroup1->ItemIndex == 0);
}

void __fastcall TMyListBoxDescendant::SubClassWndProc(Messages::TMessage &Message)
{
  if (Message.Msg == WM_STYLEMESSAGE)
    Style = (TListBoxStyle)Message.WParam;
  else
    WndProc(Message);
}

void __fastcall TMyListBoxDescendant::ToggleSubClass(bool On)
{
  if (On)
    WindowProc = SubClassWndProc;
  else
    WindowProc = WndProc;
}

#include <memory>       //for STL auto_ptr class

void __fastcall TMyListBoxDescendant::OnDrawItemProc(TWinControl *Control, int Index,
      const TRect &Rect, TOwnerDrawState State)
{
  Graphics::TBitmap *bitmap; // temporary variable for item's bitmap
  int Offset = 2;   // default text offset width
  // note that we draw on the listbox's canvas, not on the form
  TCanvas *canvas = dynamic_cast<TListBox *>(Control)->Canvas;
  canvas->FillRect(Rect); // clear the rectangle
  bitmap = dynamic_cast<Graphics::TBitmap *>((dynamic_cast<TListBox
*>(Control))->Items->Objects[Index]);
  if (bitmap)
  {
    canvas->BrushCopy(
      Bounds(Rect.Left + Offset, Rect.Top, bitmap->Width, bitmap->Height),
      bitmap, Bounds(0, 0, bitmap->Width, bitmap->Height), clRed); // render bitmap
    Offset += bitmap->Width + 4;   // add four pixels between bitmap and text
  }
  // display the text
  canvas->TextOut(Rect.Left + Offset, Rect.Top, dynamic_cast<TListBox
*>(Control)->Items->Strings[Index]);
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  MyListBoxDescendant1 = new TMyListBoxDescendant(Form1); // The owner cleans this up.
  MyListBoxDescendant1->Visible = True;
  MyListBoxDescendant1->Parent = Form1;
```

```
  MyListBoxDescendant1->Visible = True;
  MyListBoxDescendant1->Left =
    SubClassRadioGroup1->Left + SubClassRadioGroup1->Width + 30;;
  MyListBoxDescendant1->Top = SubClassRadioGroup1->Top;
  MyListBoxDescendant1->Height = SubClassRadioGroup1->Height;
  MyListBoxDescendant1->OnDrawItem =
    MyListBoxDescendant1->OnDrawItemProc;

  static std::auto_ptr<Graphics::TBitmap> _bitmap0Cleaner(bitmap0 = new Graphics::TBitmap);
  ImageList1->GetBitmap(0, bitmap0);
  MyListBoxDescendant1->Items->AddObject("Butterfly", bitmap0);

  SubClassRadioGroup1->Items->Add("SubClassWndProc");
  SubClassRadioGroup1->Items->Add("WndProc");
  SubClassRadioGroup1->ItemIndex = 2;
}
```

Delphi Examples:

```
{
This example shows how to use the WndProc method and the
WindowProc property to subclass a custom control's window
procedure.  This example subclasses the window procedure of
a TListBox descendant to respond to a user-defined message
called WM_STYLEMESSAGE. The subclassed window procedure can
be turned on or off by pressing a radio button.
}

type
  TMyListBoxDescendant = class(TlistBox)
    procedure SubClassWndProc(var Message: TMessage);
    procedure ToggleSubClass(On: Boolean);
    procedure OnDrawItemProc(
      Control: TWinControl;
      Index: Integer;
      Rect:TRect;
      State: TOwnerDrawState);
  end;
  TForm1 = class(TForm)
    SubClassRadioGroup1: TRadioGroup;
    Button1: TButton;
    ImageList1: TImageList;
    Button2: TButton;
    procedure SubClassRadioGroup1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  MyListBoxDescendant1: TMyListBoxDescendant;
  bitmap0: TBitmap;

implementation

{$R *.dfm}

const WM_STYLEMESSAGE = WM_USER + 2000;

procedure TForm1.Button1Click(Sender: TObject);
begin
```

```
    PostMessage(
      MyListBoxDescendant1.Handle,
      WM_STYLEMESSAGE,
      Integer(lbOwnerDrawFixed),
      0);
  end;

  procedure TForm1.Button2Click(Sender: TObject);
  begin
    PostMessage(
      MyListBoxDescendant1.Handle,
      WM_STYLEMESSAGE,
      Integer(lbStandard),
      0);
  end;

  procedure TMyListBoxDescendant.SubClassWndProc(var Message: TMessage);
  begin
    if (Message.Msg = WM_STYLEMESSAGE) then
      Style:= TListBoxStyle(Message.WParam)
    else
      WndProc(Message);
  end;

  procedure TMyListBoxDescendant.ToggleSubClass(On: Boolean);
  begin
    if On then
      WindowProc := SubClassWndProc
    else
      WindowProc := WndProc;
  end;

  procedure TForm1.SubClassRadioGroup1Click(Sender: TObject);
  begin
    MyListBoxDescendant1.ToggleSubClass(
      SubClassRadioGroup1.ItemIndex = 0);
  end;

  procedure TForm1.FormCreate(Sender: TObject);
  begin
    MyListBoxDescendant1:= TMyListBoxDescendant.Create(self);
    MyListBoxDescendant1.Visible:= True;
    MyListBoxDescendant1.Parent:= Form1;
    MyListBoxDescendant1.Visible:= True;
    MyListBoxDescendant1.Left:=
      SubClassRadioGroup1.Left + SubClassRadioGroup1.Width + 30;;
    MyListBoxDescendant1.Top:= SubClassRadioGroup1.Top;
    MyListBoxDescendant1.Height:= SubClassRadioGroup1.Height;
    MyListBoxDescendant1.OnDrawItem:=
      MyListBoxDescendant1.OnDrawItemProc;

    bitmap0 := TBitmap.Create;
    ImageList1.GetBitmap(0, bitmap0);
    MyListBoxDescendant1.Items.AddObject('Butterfly', bitmap0);

    SubClassRadioGroup1.Items.Add('SubClassWndProc');
    SubClassRadioGroup1.Items.Add('WndProc');
    SubClassRadioGroup1.ItemIndex := 2;
  end;

  procedure TMyListBoxDescendant.OnDrawItemProc(
    Control: TWinControl;
    Index: Integer;
    Rect:TRect;
    State: TOwnerDrawState);
  var
```

```
    Bitmap: TBitmap;      { temporary variable for the item's bitmap }
    Offset: Integer;      { text offset width }
  begin
    { draw on control canvas, not on the form }
    with (Control as TListBox).Canvas do
    begin
      FillRect(Rect);       { clear the rectangle }
      Offset := 2;          { provide default offset }
      Bitmap := TBitmap((Control as TListBox).Items.Objects[Index]);  { get the bitmap }
      if Bitmap <> nil then
      begin
        BrushCopy(
          Bounds(Rect.Left + Offset, Rect.Top, Bitmap.Width, Bitmap.Height),
          Bitmap,
          Bounds(0, 0, Bitmap.Width, Bitmap.Height),
          clRed);  {render bitmap}
        Offset := Bitmap.width + 6;    { add four pixels between bitmap and text}
      end;
      TextOut(Rect.Left + Offset, Rect.Top, (Control as TListBox).Items[Index])  { display the
  text }
    end;
  end;
```

See Also

DefWndProc

MainWndProc

WndProc

WindowProc

48.1.4.5 TOleControl Properties

The properties of the TOleControl class are listed here.

Protected Properties

	Name	Description
	ControlData (see page 2696)	Specifies the PControlData (see page 2704) associated with TOleControl (see page 2638).

Public Properties

	Name	Description
	Anchors (see page 2695)	Specifies how the control is anchored to its parent.
R	DefaultDispatch (see page 2696)	The default dispatch interface.
R	OleObject (see page 2697)	The ActiveX control associated with the TOleControl (see page 2638) object.
R	PerPropBrowsing (see page 2697)	Provides access to information about the property page properties of the OleControl.
	TabStop (see page 2697)	Determines if the user can tab to a control.

48.1.4.5.1 TOleControl.Anchors Property

Specifies how the control is anchored to its parent.

Pascal

```
property Anchors;
```

C++

```
__property Anchors;
```

Description

Use Anchors to ensure that a control maintains its current position relative to an edge of its parent, even if the parent is resized. When its parent is resized, the control holds its position relative to the edges to which it is anchored.

If a control is anchored to opposite edges of its parent, the control stretches when its parent is resized. For example, if a control has its Anchors property set to [akLeft, akRight], the control stretches when the width of its parent changes.

Anchors is enforced only when the parent is resized. Thus, for example, if a control is anchored to opposite edges of a form at design time and the form is created in a maximized state, the control is not stretched because the form is not resized after the control is created.

Note: If a control should maintain contact with three edges of its parent (hugging one side of the parent and stretching the length of that side), use the Align property instead. Unlike Anchors, Align allows controls to adjust to changes in the size of other aligned sibling controls as well as changes to the parent's size.

See Also

Parent

Align

48.1.4.5.2 TOleControl.ControlData Property

Specifies the PControlData (see page 2704) associated with TOleControl (see page 2638).

Pascal

```
property ControlData: PControlData;
```

C++

```
__property PControlData ControlData;
```

Description

Use ControlData to get or set the PControlData (see page 2704) associated with TOleControl (see page 2638). The returned value is a pointer to a TControlData (see page 2701) record, which contains all the information associated to the ActiveX control.

See Also

TControlData (see page 2701)

PControlData (see page 2704)

48.1.4.5.3 TOleControl.DefaultDispatch Property

The default dispatch interface.

Pascal

```
property DefaultDispatch: IDispatch;
```

C++

```
__property IDispatch DefaultDispatch;
```

Description

DefaultDispatch default dispatch interface that the TOleControl (see page 2638) object encapsulates. For those ActiveX controls that have multiple dispatch interfaces, one is typically identified as the default.

48.1.4.5.4 TOleControl.OleObject Property

The ActiveX control associated with the TOleControl (see page 2638) object.

Pascal

```
property OleObject: Variant;
```

C++

```
__property Variant OleObject;
```

Description

OleObject is used to get the ActiveX control that is associated with the TOleControl (see page 2638)

48.1.4.5.5 TOleControl.PerPropBrowsing Property

Provides access to information about the property page properties of the OleControl.

Pascal

```
property PerPropBrowsing: IPerPropertyBrowsing;
```

C++

```
__property IPerPropertyBrowsing PerPropBrowsing;
```

Description

PerPropBrowsing is an interface to information about the properties of the OleControl as they appear on the associated property page.

See Also

BrowseProperties (see page 2649)

48.1.4.5.6 TOleControl.TabStop Property

Determines if the user can tab to a control.

Pascal

```
property TabStop;
```

C++

```
__property TabStop;
```

Description

Use the TabStop to allow or disallow access to the control using the Tab key.

If TabStop is true, the control is in the tab order. If TabStop is false, the control is not in the tab order and the user can't press the Tab key to move to the control.

Note: TabStop is not meaningful for a form unless the form assigns another form to be its parent.

See Also

TabOrder

48.1.4.6 TOleControl Events

The events of the TOleControl class are listed here.

Public Events

	Name	Description
	ServiceQuery (see page 2698)	Specifies the TServiceQuery (see page 2706) associated with this ActiveX control.

48.1.4.6.1 TOleControl.ServiceQuery Event

Specifies the TServiceQuery (see page 2706) associated with this ActiveX control.

Pascal

```
property ServiceQuery: TServiceQuery;
```

C++

```
__property TServiceQuery ServiceQuery;
```

Description

Use TServiceQuery (see page 2706) to set or get the TServiceQuery (see page 2706) used by this ActiveX control. The value of ServiceQuery is used by the QueryService (see page 2675) method. Whenever QueryService (see page 2675) is called, it forwards the call to the associated TServiceQuery (see page 2706).

See Also

QueryService (see page 2675)

48.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	EmptyParam (see page 2699)	This is function OleCtrls.EmptyParam.
	FontToOleFont (see page 2699)	Returns a Variant containing an IFontDispatch interface that represents a TFont object.
	OleFontToFont (see page 2700)	Fills in a TFont structure to represent an IFontDispatch.

48.2.1 OleCtrls.EmptyParam Function

Pascal

```
function EmptyParam: OleVariant;
```

C++

```
OleVariant EmptyParam();
```

File

OleCtrls

Description

This is function OleCtrls.EmptyParam.

48.2.2 OleCtrls.FontToOleFont Function

Returns a Variant containing an IFontDispatch interface that represents a TFont object.

Pascal

```
function FontToOleFont(Font: TFont): Variant;
```

C++

```
Variant FontToOleFont(TFont Font);
```

File

OleCtrls

Description

Use FontToOleFont to convert a font from TFont to IFontDispatch format. The interface uses the TFont object to implement the methods of the interface.

See Also

OleFontToFont (see page 2700)

OleStrToStrVar

StringToOleStr

OleStrToString

48.2.3 OleCtrls.OleFontToFont Function

Fills in a TFont structure to represent an IFontDispatch.

Pascal

```
procedure OleFontToFont(const OleFont: Variant; Font: TFont);
```

C++

```
OleFontToFont(const Variant OleFont, TFont Font);
```

File

OleCtrls

Description

Use OleFontToFont to convert a font from COM (IFontDispatch) to TFont format.

See Also

FontToOleFont (see page 2699)

OleStrToStrVar

StringToOleStr

OleStrToString

48.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Records

	Name	Description
	TControlData (see page 2701)	This is record OleCtrls.TControlData.
	TControlData2 (see page 2702)	This is record OleCtrls.TControlData2.
	TEnumValue (see page 2702)	This is record OleCtrls.TEnumValue.

48.3.1 OleCtrls.TControlData Record

Pascal

```
TControlData = record
  ClassID: TGUID;
  EventIID: TGUID;
  EventCount: Longint;
  EventDispIDs: Pointer;
  LicenseKey: Pointer;
  Flags: DWORD;
  Version: Integer;
  FontCount: Integer;
  FontIDs: PDispIDList;
  PictureCount: Integer;
  PictureIDs: PDispIDList;
  Reserved: Integer;
  InstanceCount: Integer;
  EnumPropDescs: TList;
end;
```

C++

```
struct TControlData {
  TGUID ClassID;
  TGUID EventIID;
  Longint EventCount;
  void * EventDispIDs;
  void * LicenseKey;
  DWORD Flags;
  int Version;
  int FontCount;
  PDispIDList FontIDs;
  int PictureCount;
  PDispIDList PictureIDs;
  int Reserved;
  int InstanceCount;
  TList EnumPropDescs;
};
```

File

OleCtrls

Description

This is record OleCtrls.TControlData.

48.3.2 OleCtrls.TControlData2 Record

Pascal

```
TControlData2 = record
  ClassID: TGUID;
  EventIID: TGUID;
  EventCount: Longint;
  EventDispIDs: Pointer;
  LicenseKey: Pointer;
  Flags: DWORD;
  Version: Integer;
  FontCount: Integer;
  FontIDs: PDispIDList;
  PictureCount: Integer;
  PictureIDs: PDispIDList;
  Reserved: Integer;
  InstanceCount: Integer;
  EnumPropDescs: TList;
  FirstEventOfs: Cardinal;
end;
```

C++

```
struct TControlData2 {
  TGUID ClassID;
  TGUID EventIID;
  Longint EventCount;
  void * EventDispIDs;
  void * LicenseKey;
  DWORD Flags;
  int Version;
  int FontCount;
  PDispIDList FontIDs;
  int PictureCount;
  PDispIDList PictureIDs;
  int Reserved;
  int InstanceCount;
  TList EnumPropDescs;
  unsigned FirstEventOfs;
};
```

File

OleCtrls

Description

This is record OleCtrls.TControlData2.

48.3.3 OleCtrls.TEnumValue Record

Pascal

```
TEnumValue = record
  Value: Longint;
  Ident: string;
end;
```

C++

```
struct TEnumValue {
  Longint Value;
  UnicodeString Ident;
};
```

File

OleCtrls

Description

This is record OleCtrls.TEnumValue.

48.4 Types

The following table lists types in this documentation.

Types

Name	Description
PControlData (see page 2704)	This is type OleCtrls.PControlData.
PControlData2 (see page 2704)	This is type OleCtrls.PControlData2.
PEnumValueList (see page 2705)	This is type OleCtrls.PEnumValueList.
TEnumValueList (see page 2705)	This is type OleCtrls.TEnumValueList.
TOleEnum (see page 2705)	Represents enumerated types that must be sent over a COM interface.
TServiceQuery (see page 2706)	This is type OleCtrls.TServiceQuery.

48.4.1 OleCtrls.PControlData Type

Pascal

```
PControlData = ^TControlData;
```

C++

```
^TControlData PControlData;
```

File

OleCtrls

Description

This is type OleCtrls.PControlData.

48.4.2 OleCtrls.PControlData2 Type

Pascal

```
PControlData2 = ^TControlData2;
```

C++

```
^TControlData2 PControlData2;
```

File

OleCtrls

Description

This is type OleCtrls.PControlData2.

48.4.3 OleCtrls.PEnumValueList Type

Pascal

```
PEnumValueList = ^TEnumValueList;
```

C++

```
^TEnumValueList PEnumValueList;
```

File

OleCtrls

Description

This is type OleCtrls.PEnumValueList.

48.4.4 OleCtrls.TEnumValueList Type

Pascal

```
TEnumValueList = array[0..32767] of TEnumValue;
```

C++

```
array[0..32767] of TEnumValue TEnumValueList;
```

File

OleCtrls

Description

This is type OleCtrls.TEnumValueList.

48.4.5 OleCtrls.TOleEnum Type

Represents enumerated types that must be sent over a COM interface.

Pascal

```
TOleEnum = ActiveX.TOleEnum;
```

C++

```
ActiveX.TOleEnum TOleEnum;
```

File

OleCtrls

Description

Dispatch interfaces do not marshal enumerated types. TOleEnum is used in the Delphi language for interface parameters that are logically an enumerated type but must be passed as an integer value instead. Typically, each interface member that uses TOleEnum defines a set of integer constants to represent the enumerated values.

48.4.6 OleCtrls.TServiceQuery Type

Pascal

```
TServiceQuery = function (const rsid, iid: TGuid; out obj): HResult of object;
```

C++

```
(const rsid, iid: TGuid; out obj): HResult ( TServiceQuery)();
```

File

OleCtrls

Description

This is type OleCtrls.TServiceQuery.

49 OleServer Namespace

This is namespace OleServer.

Topics

Name	Description
Classes (see page 2708)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 2725)	The following table lists structs, records, enums in this documentation.
Types (see page 2727)	The following table lists types in this documentation.

Classes

	Name	Description
	TOleServer (see page 2708)	TOleServer is the base class for components that represent an imported COM server.
	TServerEventDispatch (see page 2720)	This is class OleServer.TServerEventDispatch.

Structs, Records, Enums

	Name	Description
	TConnectKind (see page 2725)	This is record OleServer.TConnectKind.
	TServerData (see page 2725)	This is record OleServer.TServerData.

Types

Name	Description
PServerData (see page 2727)	This is type OleServer.PServerData.
TVariantArray (see page 2727)	This is type OleServer.TVariantArray.

49.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TOleServer (see page 2708)	TOleServer is the base class for components that represent an imported COM server.
	TServerEventDispatch (see page 2720)	This is class OleServer.TServerEventDispatch.

49.1.1 TOleServer Class

TOleServer is the base class for components that represent an imported COM server.

Class Hierarchy

Pascal

```
TOleServer = class(TComponent, IUnknown);
```

C++

```
class TOleServer : public TComponent, public IUnknown;
```

File

OleServer

Description

TOleServer is the base class for COM servers that are imported by choosing Component | Import Component from the IDE. TOleServer introduces several properties and methods that manage the connection to a COM server and for invoking its default events.

Descendants of TOleServer expose the properties, methods, and events of the server CoClass. When an application calls one of the TOleServer descendant's methods or sets one of its properties, the component automatically establishes a connection to the COM server. You can also connect explicitly by calling the Connect (see page 2712) method.

Do not create instances of TOleServer. TOleServer relies on abstract pure virtual methods to establish and break a connection to the COM server. Instead, use a descendant of TOleServer that wraps the desired server. Numerous TOleServer descendants are provided on the Servers tab of the Component Palette. In addition, you can create your own descendants by importing a server's type library.

See Also

Code Generated When You Import Type Library Information

49.1.1.1 TOleServer Members

The following tables list the members exposed by TOleServer.

Protected Methods

	Name	Description
	_AddRef (see page 2711)	Increments the reference count for the number of interface references dependent on the OLE server.
	_Release (see page 2712)	Decrements the reference count for the calling interface on a object.
	ConnectEvents (see page 2713)	Establishes a connection using the EventDispatch (see page 2718) property as an event sink.
	DisconnectEvents (see page 2713)	Terminates a connection that was established using the ConnectEvents (see page 2713) method.
	GetAutoConnect (see page 2714)	Returns the value of the AutoConnect (see page 2717) property.
	GetConnectKind (see page 2714)	Returns the value of the ConnectKind (see page 2718) property.
	GetServer (see page 2714)	Returns an interface for the COM server.
	InitServerData (see page 2715)	Initializes the ServerData (see page 2719) property to reflect the type library information of the COM server.
	InvokeEvent (see page 2715)	Dispatches an event from the COM server to the appropriate event handler.
	Loaded (see page 2716)	Connects to the COM server when the Loaded object is first loaded into memory if AutoConnect (see page 2717) is True true.
	QueryInterface (see page 2716)	Returns a reference to a specified interface if the OLE server supports that interface.
	SetAutoConnect (see page 2716)	Sets the value of the AutoConnect (see page 2717) property.
	SetConnectKind (see page 2717)	Sets the value of the ConnectKind (see page 2718) property.

Protected Properties

	Name	Description
	EventDispatch (see page 2718)	Specifies a helper object that enables the TOleServer (see page 2708) object to handle server events.
	ServerData (see page 2719)	Encodes information about the COM server to which ServerData connects.

Public Methods

	Name	Description
	Connect (see page 2712)	Establishes a connection to the COM server.
	Create (see page 2709)	Allocates memory and constructs a safely initialized instance of a component.
	Destroy (see page 2710)	Disposes of the component and its owned components.
	Disconnect (see page 2713)	Terminates the connection to the COM server.

Published Properties

	Name	Description
	AutoConnect (see page 2717)	Specifies whether the component automatically connects to the server.
	ConnectKind (see page 2718)	Indicates the type of process to which the ConnectKind component establishes a connection.
	RemoteMachineName (see page 2719)	Specifies the machine running the COM server when ConnectKind (see page 2718) is ckRemote.

49.1.1.2 TOleServer.Create Constructor

Allocates memory and constructs a safely initialized instance of a component.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TOleServer(TComponent * AOwner);
```

Description

All objects have a Create method that constructs the object. TComponent redefines Create so that, for components, Create also:

establishes the relationship of a component and its Owner, as indicated by the AOwner parameter

sets the ComponentStyle property to csInheritable, meaning that the component can be inherited by a descendant form type

It is not necessary to explicitly create components added in the form designer. These components are created automatically when the application is run, and they are destroyed when the application is closed.

For components created programmatically, that is, not created in the form designer, call Create and pass in an owner component as the AOwner parameter. The owner disposes of the component when it is destroyed. If the component is not owned, then use Free when it needs to be destroyed.

Tip: When passing in Self as the Owner parameter, consider what Self references. If a component creates another component in one of its methods, then Self refers to the first component and not the component being created, which is then owned by the first component.

Note: The TComponent constructor is virtual in part to allow polymorphic instantiation of class references. This is critical to the streaming system and to the form designer. Do not forget to use the override directive when declaring a new component's Create constructor.

See Also

Components

ComponentStyle

System::TObject::Create

Destroy

System::TObject::Free

Owner

49.1.1.3 TOleServer.Destroy Destructor

Disposes of the component and its owned components.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TOleServer();
```

Description

Do not call Destroy directly. Call Free instead. Free verifies that the component is not nil, and only then calls Destroy.

Never explicitly free a component in one of its own event handlers, nor free a component from the event handler of a component that it owns or contains.

To destroy a form, call its Release method. Release waits for all the form's event handlers and the event handlers of the form's components to finish executing before destroying the form.

Note: A form owns all the controls and non-visual components that are placed on it in design mode. When it is freed, all of these components are automatically freed as well. By default, all forms are owned by the global Application object. When an application terminates, it frees the global Application object, which frees all forms. For objects that are not components, and for components created with a nil owner, be sure to call Free after finishing with the object; otherwise the memory allocated for the object will be lost until the application terminates.

See Also

System::TObject::Destroy

DestroyComponents

Destroying

System::TObject::Free

Notification

Owner

Release

Components

49.1.1.4 TOleServer Methods

The methods of the TOleServer class are listed here.

Protected Methods

	Name	Description
	_AddRef (see page 2711)	Increments the reference count for the number of interface references dependent on the OLE server.
	_Release (see page 2712)	Decrements the reference count for the calling interface on a object.
	ConnectEvents (see page 2713)	Establishes a connection using the EventDispatch (see page 2718) property as an event sink.
	DisconnectEvents (see page 2713)	Terminates a connection that was established using the ConnectEvents (see page 2713) method.
	GetAutoConnect (see page 2714)	Returns the value of the AutoConnect (see page 2717) property.
	GetConnectKind (see page 2714)	Returns the value of the ConnectKind (see page 2718) property.
V	GetServer (see page 2714)	Returns an interface for the COM server.
A	InitServerData (see page 2715)	Initializes the ServerData (see page 2719) property to reflect the type library information of the COM server.
V	InvokeEvent (see page 2715)	Dispatches an event from the COM server to the appropriate event handler.
V	Loaded (see page 2716)	Connects to the COM server when the Loaded object is first loaded into memory if AutoConnect (see page 2717) is True true.
V	QueryInterface (see page 2716)	Returns a reference to a specified interface if the OLE server supports that interface.
	SetAutoConnect (see page 2716)	Sets the value of the AutoConnect (see page 2717) property.
	SetConnectKind (see page 2717)	Sets the value of the ConnectKind (see page 2718) property.

Public Methods

	Name	Description
A	Connect (see page 2712)	Establishes a connection to the COM server.
A	Disconnect (see page 2713)	Terminates the connection to the COM server.

49.1.1.4.1 TOleServer._AddRef Method

Increments the reference count for the number of interface references dependent on the OLE server.

Pascal

```
function _AddRef: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _AddRef();
```

Description

_AddRef implements the IUnknown interface method, AddRef, which increments the reference count of objects using the OLE server's interface. _AddRef returns the resulting value of the reference count.

See Also

Invoke

QueryInterface (see page 2716)

_Release (see page 2712)

49.1.1.4.2 TOleServer._Release Method

Decrements the reference count for the calling interface on a object.

Pascal

```
function _Release: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _Release();
```

Description

_Release implements the IUnknown interface method, Release, which decrements the reference count of objects using the OLE server's interface. If the reference count on the interface falls to 0, the server object is freed from memory.

_Release returns the resulting value of the reference count.

See Also

Invoke

QueryInterface (see page 2716)

_AddRef (see page 2711)

49.1.1.4.3 TOleServer.Connect Method

Establishes a connection to the COM server.

Pascal

```
procedure Connect; virtual; abstract;
```

C++

```
virtual __fastcall Connect() = 0;
```

Description

Call Connect to connect the Connect component to the COM server it represents. If the AutoConnect (see page 2717) property is True true, the Connect component automatically calls Connect when the application is first run.

Do not use Connect if the value of ConnectKind (see page 2718) is ckAttachToInterface. When attaching to an interface, use the ConnectTo method, which is introduced by descendant classes, instead.

If the value of ConnectKind (see page 2718) is ckRemote, the RemoteMachineName (see page 2719) property must be set before calling Connect.

Connect introduces Connect as an abstract a pure virtual method. Descendant classes must override this method to let the Connect component connect to a specific type of COM server.

See Also

AutoConnect (see page 2717)

ConnectKind (see page 2718)

RemoteMachineName (see page 2719)

Disconnect (see page 2713)

Using Component Wrappers

49.1.1.4.4 TOleServer.ConnectEvents Method

Establishes a connection using the EventDispatch (see page 2718) property as an event sink.

Pascal

```
procedure ConnectEvents(const Obj: IUnknown);
```

C++

```
__fastcall ConnectEvents(const IUnknown Obj);
```

Description

ConnectEvents descendants use ConnectEvents internally to implement their event handlers for events on the COM server. If the ConnectEvents descendant does not expose any server events, ConnectEvents is not used.

The Obj parameter is an interface for the COM server that generates events.

See Also

EventDispatch (see page 2718)

DisconnectEvents (see page 2713)

49.1.1.4.5 TOleServer.Disconnect Method

Terminates the connection to the COM server.

Pascal

```
procedure Disconnect; virtual; abstract;
```

C++

```
virtual __fastcall Disconnect() = 0;
```

Description

Call Disconnect to terminate a connection formed using the Connect (see page 2712) method.

Disconnect introduces Disconnect as an abstract a pure virtual method. Descendant classes must override this method to let the Disconnect component disconnect from a specific type of COM server.

See Also

Connect (see page 2712)

Using Component Wrappers

49.1.1.4.6 TOleServer.DisconnectEvents Method

Terminates a connection that was established using the ConnectEvents (see page 2713) method.

Pascal

```
procedure DisconnectEvents(const Obj: Iunknown);
```

C++

```
__fastcall DisconnectEvents(const Iunknown Obj);
```

Description

DisconnectEvents descendants use DisconnectEvents internally to terminate the connection established by ConnectEvents (see page 2713). If the DisconnectEvents descendant does not expose any server events, DisconnectEvents is not used.

The Obj parameter is an interface for the COM server that generates events.

See Also

ConnectEvents (see page 2713)

49.1.1.4.7 TOleServer.GetAutoConnect Method

Returns the value of the AutoConnect (see page 2717) property.

Pascal

```
function GetAutoConnect: Boolean;
```

C++

```
__fastcall Boolean GetAutoConnect();
```

Description

GetAutoConnect is the protected read implementation for the AutoConnect (see page 2717) property.

See Also

AutoConnect (see page 2717)

49.1.1.4.8 TOleServer.GetConnectKind Method

Returns the value of the ConnectKind (see page 2718) property.

Pascal

```
function GetConnectKind: TConnectKind;
```

C++

```
__fastcall TConnectKind GetConnectKind();
```

Description

GetConnectKind is the protected read implementation for the ConnectKind (see page 2718) property.

See Also

ConnectKind (see page 2718)

49.1.1.4.9 TOleServer.GetServer Method

Returns an interface for the COM server.

Pascal

```
function GetServer: IUnknown; virtual;
```

C++

```
virtual __fastcall IUnknown GetServer();
```

Description

GetServer descendants call GetServer from the implementation of their Connect (see page 2712) methods. GetServer obtains an interface for the COM server described by ServerData (see page 2719). The ConnectKind (see page 2718) property indicates how GetServer can obtain the interface.

See Also

Connect (see page 2712)

ConnectKind (see page 2718)

ServerData (see page 2719)

49.1.1.4.10 TOleServer.InitServerData Method

Initializes the ServerData (see page 2719) property to reflect the type library information of the COM server.

Pascal

```
procedure InitServerData; virtual; abstract;
```

C++

```
virtual __fastcall InitServerData() = 0;
```

Description

The InitServerData constructor calls InitServerData to initialize ServerData (see page 2719) with information from the COM server's type library.

InitServerData is an abstract a pure virtual method in InitServerData. Each InitServerData descendant implements this method so that the ServerData (see page 2719) property reflects its particular COM server.

See Also

ServerData (see page 2719)

Create (see page 2709)

49.1.1.4.11 TOleServer.InvokeEvent Method

Dispatches an event from the COM server to the appropriate event handler.

Pascal

```
procedure InvokeEvent(DispID: TDispID; var Params: TVariantArray); virtual;
```

C++

```
virtual __fastcall InvokeEvent(TDispID DispID, TVariantArray Params);
```

Description

InvokeEvent descendants use InvokeEvent internally to link their event handlers to events on the COM server.

DispID is the dispatch ID for the appropriate event handler.

Params passes the parameters of the event handler as an array of OleVariants.

As implemented in InvokeEvent, InvokeEvent does nothing, because InvokeEvent has no events. Descendants that surface events from a COM server override this method so that the event sink specified by EventDispatch (see page 2718) can pass

events on to the appropriate event handlers.

See Also

EventDispatch (see page 2718)

49.1.1.4.12 TOleServer.Loaded Method

Connects to the COM server when the Loaded object is first loaded into memory if AutoConnect (see page 2717) is True true.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

The VCL streaming system automatically calls Loaded after the Loaded component is loaded into memory.

Loaded checks the AutoConnect (see page 2717) property, and if it is True true, calls the Connect (see page 2712) method to establish a connection.

See Also

AutoConnect (see page 2717)

Loaded

49.1.1.4.13 TOleServer.QueryInterface Method

Returns a reference to a specified interface if the OLE server supports that interface.

Pascal

```
function QueryInterface(const IID: TGUID; out Obj): HResult; override;
```

C++

```
virtual __fastcall HRESULT QueryInterface(const TGUID IID,  Obj);
```

Description

QueryInterface implements the IUnknown interface method QueryInterface. QueryInterface returns a reference to the interface specified by the IID parameter as the Obj parameter. If the OLE server does not support the interface, the Obj parameter returns nil NULL and QueryInterface returns a nonzero error code.

See Also

_AddRef (see page 2711)

_Release (see page 2712)

Invoke

49.1.1.4.14 TOleServer.SetAutoConnect Method

Sets the value of the AutoConnect (see page 2717) property.

Pascal

```
procedure SetAutoConnect(flag: Boolean);
```

C++

```
__fastcall SetAutoConnect(Boolean flag);
```

Description

SetAutoConnect is the protected write implementation for the AutoConnect (see page 2717) property.

See Also

AutoConnect (see page 2717)

49.1.1.4.15 TOleServer.SetConnectKind Method

Sets the value of the ConnectKind (see page 2718) property.

Pascal

```
procedure SetConnectKind(ck: TConnectKind);
```

C++

```
__fastcall SetConnectKind(TConnectKind ck);
```

Description

SetConnectKind is the protected write implementation for the ConnectKind (see page 2718) property.

See Also

ConnectKind (see page 2718)

49.1.1.5 TOleServer Properties

The properties of the TOleServer class are listed here.

Protected Properties

	Name	Description
	EventDispatch (see page 2718)	Specifies a helper object that enables the TOleServer (see page 2708) object to handle server events.
	ServerData (see page 2719)	Encodes information about the COM server to which ServerData connects.

Published Properties

	Name	Description
	AutoConnect (see page 2717)	Specifies whether the component automatically connects to the server.
	ConnectKind (see page 2718)	Indicates the type of process to which the ConnectKind component establishes a connection.
	RemoteMachineName (see page 2719)	Specifies the machine running the COM server when ConnectKind (see page 2718) is ckRemote.

49.1.1.5.1 TOleServer.AutoConnect Property

Specifies whether the component automatically connects to the server.

Pascal

```
property AutoConnect: Boolean;
```

C++

```
__property Boolean AutoConnect;
```

Description

Set AutoConnect to True true to specify whether the server should be loaded automatically when the application first starts up. If AutoConnect is False false, the application establishes a connection when it calls one of the AutoConnect descendant's methods, sets a property, or calls the Connect (see page 2712) method.

Note: AutoConnect only establishes a connection when the application is first run. Changing the AutoConnect property at runtime has no effect.

See Also

Connect (see page 2712)

Using Component Wrappers

49.1.1.5.2 TOleServer.ConnectKind Property

Indicates the type of process to which the ConnectKind component establishes a connection.

Pascal

```
property ConnectKind: TConnectKind;
```

C++

```
__property TConnectKind ConnectKind;
```

Description

Set ConnectKind to indicate how the ConnectKind component establishes a connection. ConnectKind establishes the connection when the application is run (if AutoConnect (see page 2717) is True true) or when the application calls the Connect (see page 2712) (or ConnectTo) method.

The following table lists the possible values:

Value	Meaning
ckRunningOrNew	Attach to a running server or create a new instance of the server.
ckNewInstance	Always create a new instance of the server.
ckRunningInstance	Only attach to a running instance of the server.
ckRemote	Bind to a remote instance of the server. When using this option, you must supply a value for RemoteMachineName (see page 2719).
ckAttachToInterface	Don't bind to the server. Instead, the application supplies an interface using the ConnectTo method, which is introduced in descendant classes. This option can't be used with the AutoConnect (see page 2717) property.

See Also

Connect (see page 2712)

RemoteMachineName (see page 2719)

AutoConnect (see page 2717)

Using Component Wrappers

49.1.1.5.3 TOleServer.EventDispatch Property

Specifies a helper object that enables the TOleServer (see page 2708) object to handle server events.

Pascal

```
property EventDispatch: TServerEventDispatch;
```

C++

```
__property TServerEventDispatch EventDispatch;
```

Description

EventDispatch is used internally by descendant objects that surface the events of a COM server. EventDispatch supports an IDispatch interface, which it uses to allow TOleServer (see page 2708) descendants to dispatch events to the appropriate event handler. TOleServer (see page 2708) introduces no events of its own. EventDispatch is used only by descendants that surface events from a COM server.

See Also

ServerData (see page 2719)

49.1.1.5.4 TOleServer.RemoteMachineName Property

Specifies the machine running the COM server when ConnectKind (see page 2718) is ckRemote.

Pascal

```
property RemoteMachineName: string;
```

C++

```
__property UnicodeString RemoteMachineName;
```

Description

Use RemoteMachineName when the COM server instance is running remotely on another machine. Specify the name of the machine on which the COM server is running as the value of RemoteMachineName.

Note: RemoteMachineName is only used if ConnectKind (see page 2718) is ckRemote. Setting RemoteMachineName to a non-empty string sets ConnectKind (see page 2718) to ckRemote.

See Also

ConnectKind (see page 2718)

Connect (see page 2712)

Using Component Wrappers

49.1.1.5.5 TOleServer.ServerData Property

Encodes information about the COM server to which ServerData connects.

Pascal

```
property ServerData: PServerData;
```

C++

```
__property PServerData ServerData;
```

Description

ServerData uses ServerData internally to store type library information describing its COM server. Each ServerData descendant initializes this property in the constructor, specifying the information it needs to establish a connection.

See Also

InitServerData (see page 2715)

49.1.2 TServerEventDispatch Class

Class Hierarchy

Pascal

```
TServerEventDispatch = class(TObject, IUnknown, IDispatch);
```

C++

```
class TServerEventDispatch : public TObject, public IUnknown, public IDispatch;
```

File

OleServer

Description

This is class OleServer.TServerEventDispatch.

49.1.2.1 TServerEventDispatch Members

The following tables list the members exposed by TServerEventDispatch.

Protected Methods

	Name	Description
	_AddRef (see page 2721)	This is _AddRef, a member of class TServerEventDispatch.
	_Release (see page 2721)	This is _Release, a member of class TServerEventDispatch.
	GetIDsOfNames (see page 2721)	GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name.
	GetTypeInfo (see page 2722)	GetTypeInfo returns the object's type information.
	GetTypeInfoCount (see page 2722)	GetTypeInfoCount returns the count of type information interfaces.
	Invoke (see page 2723)	Invoke performs a call to a late-binded property or method.
	QueryInterface (see page 2723)	This is QueryInterface, a member of class TServerEventDispatch.
	ServerDisconnect (see page 2723)	This is ServerDisconnect, a member of class TServerEventDispatch.

Protected Properties

	Name	Description
R	Server (see page 2724)	This is Server, a member of class TServerEventDispatch.

Public Methods

	Name	Description
	Create (see page 2720)	This is Create, a member of class TServerEventDispatch.

49.1.2.2 TServerEventDispatch.Create Constructor

Pascal

```
constructor Create(Server: TOleServer);
```

C++

```
__fastcall TServerEventDispatch(TOleServer Server);
```

Description

This is Create, a member of class TServerEventDispatch.

49.1.2.3 TServerEventDispatch Methods

The methods of the TServerEventDispatch class are listed here.

Protected Methods

	Name	Description
	_AddRef (see page 2721)	This is _AddRef, a member of class TServerEventDispatch.
	_Release (see page 2721)	This is _Release, a member of class TServerEventDispatch.
	GetIDsOfNames (see page 2721)	GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name.
	GetTypeInfo (see page 2722)	GetTypeInfo returns the object's type information.
	GetTypeInfoCount (see page 2722)	GetTypeInfoCount returns the count of type information interfaces.
	Invoke (see page 2723)	Invoke performs a call to a late-binded property or method.
	QueryInterface (see page 2723)	This is QueryInterface, a member of class TServerEventDispatch.
	ServerDisconnect (see page 2723)	This is ServerDisconnect, a member of class TServerEventDispatch.

49.1.2.3.1 TServerEventDispatch._AddRef Method

Pascal

```
function _AddRef: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _AddRef();
```

Description

This is _AddRef, a member of class TServerEventDispatch.

49.1.2.3.2 TServerEventDispatch._Release Method

Pascal

```
function _Release: Integer; stdcall;
```

C++

```
__fastcall __stdcall int _Release();
```

Description

This is _Release, a member of class TServerEventDispatch.

49.1.2.3.3 TServerEventDispatch.GetIDsOfNames Method

GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name.

Pascal

```
function GetIDsOfNames(const IID: TGUID; Names: Pointer; NameCount: Integer; LocaleID:
Integer; DispIDs: Pointer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetIDsOfNames(const TGUID IID, void * Names, int NameCount, int
LocaleID, void * DispIDs);
```

Description

GetIDsOfNames returns a set of dispatch identifiers (**dispid**s) for a given member name. GetIDsOfNames is used primarily to obtain dispatch identifiers for the late-binded methods exposed by the COM object through the IDispatch interface.

See Also

GetTypeInfoCount

GetTypeInfo

Invoke

49.1.2.3.4 TServerEventDispatch.GetTypeInfo Method

GetTypeInfo returns the object's type information.

Pascal

```
function GetTypeInfo(Index: Integer; LocaleID: Integer; out TypeInfo): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetTypeInfo(int Index, int LocaleID,  TypeInfo);
```

Description

GetTypeInfo returns the type information for a specified index. The index is previously obtained by a call to the GetTypeInfoCount method. If the method returns S_OK, the `TypeInfo` parameter is set to point to the requested type information; otherwise an error code is returned.

See Also

GetTypeInfoCount

GetIDsOfNames

Invoke

49.1.2.3.5 TServerEventDispatch.GetTypeInfoCount Method

GetTypeInfoCount returns the count of type information interfaces.

Pascal

```
function GetTypeInfoCount(out Count: Integer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetTypeInfoCount(int Count);
```

Description

GetTypeInfoCount returns the count of type information interfaces supported by the implementing COM object. If the method returns S_OK, the `Count` parameter is set to the number of type information interfaces, which is either 1 (if the object supports type information) or 0 (if the object does not support type information). A return value of E_NOTIMPL means that the method is not implemented in the COM object.

See Also

GetTypeInfo

GetIDsOfNames

Invoke

49.1.2.3.6 TServerEventDispatch.Invoke Method

Invoke performs a call to a late-binded property or method.

Pascal

```
function Invoke(DispID: Integer; const IID: TGUID; LocaleID: Integer; Flags: Word; var Params;
VarResult: Pointer; ExcepInfo: Pointer; ArgErr: Pointer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT Invoke(int DispID, const TGUID IID, int LocaleID, Word Flags,
Params, void * VarResult, void * ExcepInfo, void * ArgErr);
```

Description

Invoke actually calls a method or a property exposed by the COM object through IDispatch. Invoke requires a previously obtained dispatch identifier. The identifier uniquely identifies a method, a property, and a list of parameters to be passed to the called method and exception information. The dispatch identifier of a method or property (also known as **dispid**) can be obtained by a calling the GetIDsOfNames method.

See Also

GetTypeInfoCount

GetTypeInfo

GetIDsOfNames

49.1.2.3.7 TServerEventDispatch.QueryInterface Method

Pascal

```
function QueryInterface(const IID: TGUID; out Obj): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT QueryInterface(const TGUID IID,  Obj);
```

Description

This is QueryInterface, a member of class TServerEventDispatch.

49.1.2.3.8 TServerEventDispatch.ServerDisconnect Method

Pascal

```
function ServerDisconnect: Boolean;
```

C++

```
__fastcall Boolean ServerDisconnect();
```

Description

This is ServerDisconnect, a member of class TServerEventDispatch.

49.1.2.4 TServerEventDispatch Properties

The properties of the TServerEventDispatch class are listed here.

Protected Properties

	Name	Description
	Server (see page 2724)	This is Server, a member of class TServerEventDispatch.

49.1.2.4.1 TServerEventDispatch.Server Property

Pascal

```
property Server: TOleServer;
```

C++

```
__property TOleServer Server;
```

Description

This is Server, a member of class TServerEventDispatch.

49.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TConnectKind (see page 2725)	This is record OleServer.TConnectKind.

Records

	Name	Description
	TServerData (see page 2725)	This is record OleServer.TServerData.

49.2.1 OleServer.TConnectKind Enumeration

Pascal

```
TConnectKind = (
  ckRunningOrNew,
  ckNewInstance,
  ckRunningInstance,
  ckRemote,
  ckAttachToInterface
);
```

C++

```
enum TConnectKind {
  ckRunningOrNew,
  ckNewInstance,
  ckRunningInstance,
  ckRemote,
  ckAttachToInterface
};
```

File

OleServer

Description

This is record OleServer.TConnectKind.

49.2.2 OleServer.TServerData Record

Pascal

```
TServerData = record
  ClassID: TGUID;
  IntfIID: TGUID;
  EventIID: TGUID;
  LicenseKey: Pointer;
  Version: Integer;
  InstanceCount: Integer;
```

```
  end;
```

C++

```
  struct TServerData {
    TGUID ClassID;
    TGUID IntfIID;
    TGUID EventIID;
    void * LicenseKey;
    int Version;
    int InstanceCount;
  };
```

File

OleServer

Description

This is record OleServer.TServerData.

49.3 Types

The following table lists types in this documentation.

Types

Name	Description
PServerData (see page 2727)	This is type OleServer.PServerData.
TVariantArray (see page 2727)	This is type OleServer.TVariantArray.

49.3.1 OleServer.PServerData Type

Pascal

```
PServerData = ^TServerData;
```

C++

```
^TServerData PServerData;
```

File

OleServer

Description

This is type OleServer.PServerData.

49.3.2 OleServer.TVariantArray Type

Pascal

```
TVariantArray = Array of OleVariant;
```

C++

```
Array of OleVariant TVariantArray;
```

File

OleServer

Description

This is type OleServer.TVariantArray.

50 OPConvert Namespace

This is namespace OPConvert.

Topics

Name	Description
Interfaces (see page 2730)	The following table lists interfaces in this documentation.
Structs, Records, Enums (see page 2740)	The following table lists structs, records, enums in this documentation.
Types (see page 2743)	The following table lists types in this documentation.

Interfaces

	Name	Description
	IOPConvert (see page 2730)	IOPConvert is the interface for marshaling and unmarshaling method calls on an invokable interface.

Structs, Records, Enums

	Name	Description
	TSOAPConvertOption (see page 2740)	TSOAPConvertOption and TSOAPConvertOptions (see page 2743) represent flags that customize the conversion between a SOAP encoding and native method calls.

Types

Name	Description
TSOAPConvertOptions (see page 2743)	This is type OPConvert.TSOAPConvertOptions.

50.1 Interfaces

The following table lists interfaces in this documentation.

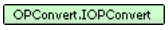
Interfaces

	Name	Description
	IOPConvert (see page 2730)	IOPConvert is the interface for marshaling and unmarshaling method calls on an invokable interface.

50.1.1 IOPConvert Interface

IOPConvert is the interface for marshaling and unmarshaling method calls on an invokable interface.

Class Hierarchy



Pascal

```
IOPConvert = interface;
```

C++

```
__interface IOPConvert;
```

File

OPConvert

Description

IOPConvert is implemented by classes that marshal and unmarshal method calls in Web Service applications. It defines methods for marshaling and unmarshaling method calls, as well as any exceptions that occur when attempting to execute those method calls.

The IOPConvert interface is used by remote interfaced objects on the client (TRIO descendants) and invoker components on the server (TSoapPascalInvoker descendants). Both the remote interfaced object and the invoker classes use the same IOPConvert implementation class, which represents a single encoding system (such as SOAP).

Classes that implement IOPConvert use the runtime type information (RTTI) on an invokable interface to interpret or encode method calls. They do not execute these calls, but rather, convert between transportable strings that are the encoding of interface method calls and an invocation context (TInvContext (see page 829) object).

Note: C++ method declarations that use IOPConvert use the _di_IOPConvert type instead. This type is a DelphiInterface wrapper around the IOPConvert interface:

```
typedef System::DelphiInterfaceDelphiInterface_object< IOPConvert >  _di_IOPConvert;
```

See Also

TOPToSoapDomConvert (see page 2747)

TRIO (see page 4056)

TSoapPascalInvoker

TInvContext (see page 829)

50.1.1.1 IOPConvert Members

The following tables list the members exposed by IOPConvert.

Public Methods

	Name	Description
	GetAttachments (see page 2732)	Returns the value of the Attachments (see page 2738) property.
	GetEncoding (see page 2732)	Returns the value of the Encoding (see page 2738) property.
	GetOptions (see page 2732)	Returns the value of the Options (see page 2739) property.
	GetTempDir (see page 2732)	Returns the value of the TempDir (see page 2739) property.
	InvContextToMsg (see page 2733)	Marshals a method call from an invocation context to a transportable string.
	MakeFault (see page 2733)	Encodes an exception that was raised when trying to execute a method on an invokable interface.
	MakeResponse (see page 2734)	Encodes the results of successfully executing a method on an invokable interface.
	MsgToInvContext (see page 2734)	This is the overview for the MsgToInvContext method overload.
	ProcessResponse (see page 2735)	This is the overview for the ProcessResponse method overload.
	SetAttachments (see page 2736)	Sets the value of the Attachments (see page 2738) property.
	SetEncoding (see page 2737)	Sets the value of the Encoding (see page 2738) property.
	SetOptions (see page 2737)	Sets the value of the Options (see page 2739) property.
	SetTempDir (see page 2737)	Sets the value of the TempDir (see page 2739) property.

Public Properties

	Name	Description
	Attachments (see page 2738)	Lists the objects that encapsulate the attachments included with the SOAP request or its response.
	Encoding (see page 2738)	Indicates the character set to use for encoded method calls.
	Options (see page 2739)	Controls how certain types of conversions are performed.
	TempDir (see page 2739)	Specifies the directory where temporary files for attachments are saved.

50.1.1.2 IOPConvert Methods

The methods of the IOPConvert class are listed here.

Public Methods

	Name	Description
	GetAttachments (see page 2732)	Returns the value of the Attachments (see page 2738) property.
	GetEncoding (see page 2732)	Returns the value of the Encoding (see page 2738) property.
	GetOptions (see page 2732)	Returns the value of the Options (see page 2739) property.
	GetTempDir (see page 2732)	Returns the value of the TempDir (see page 2739) property.
	InvContextToMsg (see page 2733)	Marshals a method call from an invocation context to a transportable string.
	MakeFault (see page 2733)	Encodes an exception that was raised when trying to execute a method on an invokable interface.
	MakeResponse (see page 2734)	Encodes the results of successfully executing a method on an invokable interface.
	MsgToInvContext (see page 2734)	This is the overview for the MsgToInvContext method overload.
	ProcessResponse (see page 2735)	This is the overview for the ProcessResponse method overload.
	SetAttachments (see page 2736)	Sets the value of the Attachments (see page 2738) property.
	SetEncoding (see page 2737)	Sets the value of the Encoding (see page 2738) property.
	SetOptions (see page 2737)	Sets the value of the Options (see page 2739) property.
	SetTempDir (see page 2737)	Sets the value of the TempDir (see page 2739) property.

50.1.1.2.1 IOPConvert.GetAttachments Method

Returns the value of the Attachments (see page 2738) property.

Pascal

```
function GetAttachments: TSoapDataList;
```

C++

```
__fastcall TSoapDataList GetAttachments();
```

Description

GetAttachments is the read implementation for the Attachments (see page 2738) property.

See Also

Attachments (see page 2738)

50.1.1.2.2 IOPConvert.GetEncoding Method

Returns the value of the Encoding (see page 2738) property.

Pascal

```
function GetEncoding: DOMString;
```

C++

```
__fastcall DOMString GetEncoding();
```

Description

GetEncoding is the read implementation for the Encoding (see page 2738) property.

See Also

Encoding (see page 2738)

50.1.1.2.3 IOPConvert.GetOptions Method

Returns the value of the Options (see page 2739) property.

Pascal

```
function GetOptions: TSOAPConvertOptions;
```

C++

```
__fastcall TSOAPConvertOptions GetOptions();
```

Description

GetOptions is the read implementation for the Options (see page 2739) property.

See Also

Options (see page 2739)

50.1.1.2.4 IOPConvert.GetTempDir Method

Returns the value of the TempDir (see page 2739) property.

Pascal

```
function GetTempDir: string;
```

C++

```
__fastcall UnicodeString GetTempDir();
```

Description

GetTempDir is the read implementation for the TempDir (see page 2739) property.

See Also

TempDir (see page 2739)

50.1.1.2.5 IOPConvert.InvContextToMsg Method

Marshals a method call from an invocation context to a transportable string.

Pascal

```
function InvContextToMsg(const IntfMD: TIntfMetaData; MethNum: Integer; Con: TInvContext;
Headers: THeaderList): TStream;
```

C++

```
__fastcall TStream InvContextToMsg(const TIntfMetaData IntfMD, int MethNum, TInvContext Con,
THeaderList Headers);
```

Description

InvContextToMsg looks up a specified method call, including the types and values of any arguments, from an invocation context, and returns an encoded version of that method call that can be transported to the application that executes it.

IntfMD represents the runtime type information (RTTI) from the interface whose method is called.

MethNum identifies the called method by its position in the interface's method table.

Con is the invocation context that contains details of the method call, including the parameter values passed in, the calling conventions, and any other temporary data used when making the call.

Headers is a list of TSOAPHeader (see page 908) descendants that represent the headers of the SOAP envelope that contained the method call.

InvContextToMsg returns a stream from which an application can read the encoded version of the method call. This encoded method call can then be passed to an IWebNode interface for execution.

See Also

MsgToInvContext (see page 2735)

Execute

50.1.1.2.6 IOPConvert.MakeFault Method

Encodes an exception that was raised when trying to execute a method on an invokable interface.

Pascal

```
procedure MakeFault(const Ex: Exception; EStream: TStream);
```

C++

```
__fastcall MakeFault(const Exception Ex, TStream EStream);
```

Description

MakeFault encodes an exception that a Web Service application catches when trying to execute a method on an invokable interface. It generates an encoded exception packet, which the client can decode and use to raise the exception on the client application.

Ex is the exception object that represents the exception that occurred.

EStream is a stream to which MakeFault writes the encoded exception packet.

See Also

MakeResponse (see page 2734)

50.1.1.2.7 IOPConvert.MakeResponse Method

Encodes the results of successfully executing a method on an invokable interface.

Pascal

```
procedure MakeResponse(const IntfMD: TIntfMetaData; const MethNum: Integer; Context:
TInvContext; Response: TStream; Headers: THeaderList);
```

C++

```
__fastcall MakeResponse(const TIntfMetaData IntfMD, const int MethNum, TInvContext Context,
TStream Response, THeaderList Headers);
```

Description

MakeResponse retrieves the results of executing a method on an invokable interface from an invocation context and packages them into a transportable string, which it writes to a stream. This string is passed back to the client, which can unpack it using the ProcessResponse (see page 2736) method.

IntfMD represents the runtime type information (RTTI) from the interface whose method was executed.

MethNum identifies the called method by its position in the interface's method table.

Context is the invocation context that contains details of the method call.

Response is a stream to which MakeResponse writes the encoded version of the method result and any output parameters.

Headers is a list of TSOAPHeader (see page 908) descendants that represent the headers that should be added to the response message.

See Also

MakeFault (see page 2733)

ProcessResponse (see page 2736)

50.1.1.2.8 MsgToInvContext Method

This is the overview for the MsgToInvContext method overload.

Overload List

	Name	Description
	IOPConvert.MsgToInvContext (InvString, TIntfMetaData, Integer, TInvContext) (see page 2735)	Unpacks an encoded method call into an invocation context so that it can be executed.

	IOPConvert.MsgToInvContext (TStream, TIntfMetaData, Integer, TInvContext, THeaderList) (see page 2735)	Unpacks an encoded method call into an invocation context so that it can be executed.

50.1.1.2.8.1 IOPConvert.MsgToInvContext Method (InvString, TIntfMetaData, Integer, TInvContext)

Unpacks an encoded method call into an invocation context so that it can be executed.

Pascal

```
procedure MsgToInvContext(const Request: InvString; const IntfMD: TIntfMetaData; var MethNum:
Integer; Context: TInvContext); overload;
procedure MsgToInvContext(const Request: TStream; const IntfMD: TIntfMetaData; var MethNum:
Integer; Context: TInvContext; Headers: THeaderList); overload;
```

C++

```
__fastcall MsgToInvContext(const InvString Request, const TIntfMetaData IntfMD, int MethNum,
TInvContext Context);
__fastcall MsgToInvContext(const TStream Request, const TIntfMetaData IntfMD, int MethNum,
TInvContext Context, THeaderList Headers);
```

Description

MsgToInvContext takes an encoded message that a server application receives and uses it to fill out an invocation context, which can then be used to execute the method call.

Request is the encoded method call, or a stream from which to read that method call. It is generated on the client by a call to InvContextToMsg (see page 2733).

IntfMD represents the runtime type information (RTTI) from the interface whose method is called.

MethNum returns the position of the called method in the interface's method table.

Context is the invocation context to which this method adds the information encoded in Request.

Headers is an object list to which MsgToInvContext adds TSOAPHeader (see page 908) descendants to represent every header in the encoded message.

See Also

InvContextToMsg (see page 2733)

50.1.1.2.9 ProcessResponse Method

This is the overview for the ProcessResponse method overload.

Overload List

	Name	Description
	IOPConvert.ProcessResponse (InvString, TIntfMetaData, TIntfMethEntry, TInvContext) (see page 2736)	Unpacks the response packet from a method call into an invocation context.
	IOPConvert.ProcessResponse (TStream, TIntfMetaData, TIntfMethEntry, TInvContext, THeaderList) (see page 2736)	Unpacks the response packet from a method call into an invocation context.

50.1.1.2.9.1 IOPConvert.ProcessResponse Method (InvString, TIntfMetaData, TIntfMethEntry, TInvContext)

Unpacks the response packet from a method call into an invocation context.

Pascal

```
procedure ProcessResponse(const Resp: TStream; const IntfMD: TIntfMetaData; const MD:
TIntfMethEntry; Context: TInvContext; Headers: THeaderList); overload;
procedure ProcessResponse(const Resp: InvString; const IntfMD: TIntfMetaData; const MD:
TIntfMethEntry; Context: TInvContext); overload; deprecated;
```

C++

```
__fastcall ProcessResponse(const TStream Resp, const TIntfMetaData IntfMD, const
TIntfMethEntry MD, TInvContext Context, THeaderList Headers);
__fastcall ProcessResponse(const InvString Resp, const TIntfMetaData IntfMD, const
TIntfMethEntry MD, TInvContext Context);
```

Description

ProcessResponse takes the encoded results of executing a method on an invokable interface from an invocation context and unpacks them into an invocation context.

Response is the encoded results of executing a method on the invokable interface, or a stream from which those results can be read. It is generated on the server by the MakeResponse (see page 2734) method if the call was successful, or the MakeFault (see page 2733) method if the call was unsuccessful.

XMLDoc is the interface to an XML document that represents the results of executing a method on the invokable interface. The values of its nodes come from the MakeResponse (see page 2734) method if the call was successful, or the MakeFault (see page 2733) method if the call was unsuccessful.

IntfMD describes the runtime type information (RTTI) of the interface whose method was called.

MD describes the runtime type information (RTTI) of the method that was called.

Context is the invocation context that ProcessResponse fills out based on the contents of Response.

Headers is a list that ProcessResponse fills with TSOAPHeader (see page 908) descendants that represent the headers of the encoded result message.

See Also

MakeFault (see page 2733)

MakeResponse (see page 2734)

50.1.1.2.10 IOPConvert.SetAttachments Method

Sets the value of the Attachments (see page 2738) property.

Pascal

```
procedure SetAttachments(Value: TSoapDataList);
```

C++

```
__fastcall SetAttachments(TSoapDataList Value);
```

Description

SetAttachments is the write implementation for the Attachments (see page 2738) property.

Value is a list of TSoapAttachment objects that encapsulate the current set of attachments.

See Also

Attachments (see page 2738)

50.1.1.2.11 IOPConvert.SetEncoding Method

Sets the value of the Encoding (see page 2738) property.

Pascal

```
procedure SetEncoding(const Encoding: DOMString);
```

C++

```
__fastcall SetEncoding(const DOMString Encoding);
```

Description

SetEncoding is the write implementation for the Encoding (see page 2738) property.

Encoding (see page 2738) is the name of the new encoding to use for method calls.

See Also

Encoding (see page 2738)

50.1.1.2.12 IOPConvert.SetOptions Method

Sets the value of the Options (see page 2739) property.

Pascal

```
procedure SetOptions(const Value: TSOAPConvertOptions);
```

C++

```
__fastcall SetOptions(const TSOAPConvertOptions Value);
```

Description

SetOptions is the write implementation for the Options (see page 2739) property.

Value is the new set of options to use when marshaling and unmarshaling method calls.

See Also

Options (see page 2739)

50.1.1.2.13 IOPConvert.SetTempDir Method

Sets the value of the TempDir (see page 2739) property.

Pascal

```
procedure SetTempDir(const Value: string);
```

C++

```
__fastcall SetTempDir(const UnicodeString Value);
```

Description

SetTempDir is the write implementation for the TempDir (see page 2739) property.

Value is the new temporary directory to use.

See Also

TempDir (see page 2739)

50.1.1.3 IOPConvert Properties

The properties of the IOPConvert class are listed here.

Public Properties

	Name	Description
	Attachments (see page 2738)	Lists the objects that encapsulate the attachments included with the SOAP request or its response.
	Encoding (see page 2738)	Indicates the character set to use for encoded method calls.
	Options (see page 2739)	Controls how certain types of conversions are performed.
	TempDir (see page 2739)	Specifies the directory where temporary files for attachments are saved.

50.1.1.3.1 IOPConvert.Attachments Property

Lists the objects that encapsulate the attachments included with the SOAP request or its response.

Pascal

```
property Attachments: TSoapDataList;
```

C++

```
__property TSoapDataList Attachments;
```

Description

When a method on an invokable interface requires the user to send or receive attachments, these attachments are each represented by a TSoapAttachment object. Attachments lists the TSoapAttachment objects for each attachment of the current method.

Note: When the application that contains IOPConvert (see page 2730) receives attachments, it writes them to a temporary file rather than storing them in memory. The location for these temporary files is specified by the TempDir (see page 2739) property.

See Also

TempDir (see page 2739)

TSOAPAttachment (see page 895)

50.1.1.3.2 IOPConvert.Encoding Property

Indicates the character set to use for encoded method calls.

Pascal

```
property Encoding: DOMString;
```

C++

```
__property DOMString Encoding;
```

Description

Use Encoding to indicate the character set that is used for encoded method calls. If you do not specify an encoding (that is, if Encoding is an empty string), UTF-8 is assumed.

See Also

DefaultDOMVendor

Encoding

50.1.1.3.3 IOPConvert.Options Property

Controls how certain types of conversions are performed.

Pascal

```
property Options: TSOAPConvertOptions;
```

C++

```
__property TSOAPConvertOptions Options;
```

Description

Set Options to indicate how you want to handle various types of conversions. This set can include zero or more of the TSOAPConvertOption (see page 2740) flags.

See Also

MsgToInvContext (see page 2735)

ClassOptions (see page 877)

InvContextToMsg (see page 2733)

50.1.1.3.4 IOPConvert.TempDir Property

Specifies the directory where temporary files for attachments are saved.

Pascal

```
property TempDir: string;
```

C++

```
__property UnicodeString TempDir;
```

Description

When the application receives an attachment as the value of a parameter (or the return value) of the current method on the invokable interface, it writes the attachment to a temporary file. This temporary file becomes the value of the CacheFile property on the TSoapAttachment object that represents the attachment.

Use TempDir to specify the directory where temporary files for attachments are written.

Note: If TempDir is not set, the application uses the default temporary directory. On Windows, this is the directory that the Windows API GetTempPath returns. On Linux, this is obtained from the TMPDIR environment setting (or, if TMPDIR is not set, /tmp/)

See Also

Attachments (see page 2738)

CacheFile (see page 904)

50.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TSOAPConvertOption (see page 2740)	TSOAPConvertOption and TSOAPConvertOptions (see page 2743) represent flags that customize the conversion between a SOAP encoding and native method calls.

50.2.1 OPConvert.TSOAPConvertOption Enumeration

TSOAPConvertOption and TSOAPConvertOptions (see page 2743) represent flags that customize the conversion between a SOAP encoding and native method calls.

Pascal

```
TSOAPConvertOption = (
  soSendUntyped,
  soSendMultiRefObj,
  soSendMultiRefArray,
  soTryAllSchema,
  soRootRefNodesToBody,
  soDocument,
  soReturnSuccessForFault,
  soUTF8InHeader,
  soDontSendEmptyNodes,
  soCacheMimeResponse,
  soDontClearOutBoundHeaders,
  soCustomFaultAtDetailsNode,
  soLiteralParams,
  soUTF8EncodeXML,
  soXXXXHdr
);
```

C++

```
enum TSOAPConvertOption {
  soSendUntyped,
  soSendMultiRefObj,
  soSendMultiRefArray,
  soTryAllSchema,
  soRootRefNodesToBody,
  soDocument,
  soReturnSuccessForFault,
  soUTF8InHeader,
  soDontSendEmptyNodes,
  soCacheMimeResponse,
  soDontClearOutBoundHeaders,
  soCustomFaultAtDetailsNode,
  soLiteralParams,
  soUTF8EncodeXML,
  soXXXXHdr
};
```

File

OPConvert

Description

TSOAPConvertOptions (see page 2743) is a set of TSOAPConvertOption values. TSOAPConvertOption values are flags that govern the conversions that TSOAPDomConv makes when translating between native code and its SOAP representation. The following table lists the possible values:

Option	Meaning
soSendUntyped	When converting from a native type to SOAP, do not include the type attribute. Some Web Service applications do not need type information to be included in the SOAP request, but simply assume that values match the function signature.
soSendMultiRefObj	When converting from a native type to SOAP, indicate when multiple references refer to the same object. That is, treat objects so that their identify is based on the object reference, rather than simply treating them as structures, where their identity is based only on the value of their properties.
soSendMultiRefArray	When converting from a native type to SOAP, indicate when multiple references refer to the same array. That is, treat arrays so that their identify is based on the array reference, rather than simply treating them as structures, where their identity is based only on the value of their elements.
soTryAllSchema	When converting from SOAP to a native type, TSOAPDomConv assumes, by default, that xml schema definitions fall in the xml schema that is associated with the application. If it is possible that the SOAP request uses another xml schema to define standard types, include soTryAllSchema. This causes TSOAPDomConv to look in all schema documents that that are defined in the xml schema unit to locate type definitions.
soRootRefNodesToBody	When converting from a native type to SOAP, multiple references all appear as children of the <soap:body> tag, rather than as siblings of the first instance. This option is recommended, because some Web Service providers require multiple references to be rooted to the body tag and will not understand the request if this flag is not included.
soDocument	When converting from SOAP to a native type, assume that the request is encoded as a document rather than as a remote procedure call (RPC). This option is used when communicating with document servers.
soReturnSuccessForFault	When the server generates a SOAP fault packet, the response message should not have an error status code, but rather it should indicate success (200). This allows servers to send fault packets to clients that generate exceptions when an HTTP response does not indicate success.
soUTF8InHeader	When converting from native types to SOAP, include a SOAP header that indicates that the SOAP packet uses a UTF8 encoding style. Because the SOAP specification does not treat UTF8 as the default encoding style, this header is technically required. Some Web Service applications, however, do not use this header.
soDontSendEmptyNodes	When converting from native types to SOAP, do not include an element for empty strings , empty arrays, or nil (Delphi) or NULL (C++) object references.
soCacheMimeResponse	When converting a response from SOAP to native types, store the incoming response content in a temporary file rather than in memory if the binding type (TWebServiceBindingType) indicates that the response uses a MIME encoding. This is because MIME encodings indicate the presence of attachments, which can be very large. Only remove this flag if the expected attachments are known to be small.
soDontClearOutBoundHeaders	When converting from native types to SOAP, do not free any classes that represent headers after generating their SOAP representation.
soCustomFaultAtDetailsNode	When generating a SOAP fault packet to represent an exception, serialize information to the details node rather than as a child of the details node.
soLiteralParams	When converting from the SOAP encoding to native types, do not unwind parameter, even if the WSDL importer was called with the option for unwinding literal parameters. This option is needed in document literal encoding when the name of a method and the name of the structure representing its parameters are different.

soUTF8EncodeXML	When converting from native types to SOAP, encode characters at the transport level to UTF-8. This encoding is less efficient than DOM-level encoding, but is required for some Web Services that assume transport-level rather than DOM-level encoding.
soXXXXHdr	The type being converted represents a header.

50.3 Types

The following table lists types in this documentation.

Types

Name	Description
TSOAPConvertOptions (see page 2743)	This is type OPConvert.TSOAPConvertOptions.

50.3.1 OPConvert.TSOAPConvertOptions Type

Pascal

```
TSOAPConvertOptions = set of TSOAPConvertOption;
```

C++

```
TSOAPConvertOption TSOAPConvertOptions;
```

File

OPConvert

Description

This is type OPConvert.TSOAPConvertOptions.

51 OPToSOAPDomConv Namespace

This is namespace OPToSOAPDomConv.

Topics

Name	Description
Classes (see page 2747)	The following table lists classes in this documentation.
Functions (see page 2783)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 2784)	The following table lists structs, records, enums in this documentation.
Types (see page 2786)	The following table lists types in this documentation.
Variables (see page 2789)	The following table lists variables in this documentation.
Constants (see page 2790)	The following table lists constants in this documentation.

Classes

	Name	Description
	ESOAPDomConvertError (see page 2747)	ESOAPDomConvertError is the class for exceptions that occur when marshaling or unmarshaling a SOAP request.
	TOPToSoapDomConvert (see page 2747)	TOPToSoapDomConvert handles the marshaling and unmarshaling of SOAP method calls.
	TSOAPDomConv (see page 2758)	TSOAPDomConv is the base class for components that use a DOM to interpret and create SOAP method calls.

Constants

Name	Description
SVarArrayType (see page 2790)	This is constant OPToSOAPDomConv.SVarArrayType.

Functions

	Name	Description
	GetOrdPropEx (see page 2783)	This is function OPToSOAPDomConv.GetOrdPropEx.

Structs, Records, Enums

	Name	Description
	ConvNodeState (see page 2784)	This is record OPToSOAPDomConv.ConvNodeState.
	TMultiRefNodeElem (see page 2784)	This is record OPToSOAPDomConv.TMultiRefNodeElem.
	TMultiRefNodeMapElem (see page 2785)	This is record OPToSOAPDomConv.TMultiRefNodeMapElem.
	TSOAPArrayElemDesc (see page 2785)	This is record OPToSOAPDomConv.TSOAPArrayElemDesc.

Types

Name	Description
TMemberDataNotReceivedEvent (see page 2786)	TMemberDataNotReceivedEvent is the type of the OnMemberDataNotReceived event of TSOAPDomConv (see page 2758).
TMultiRefNodeMap (see page 2787)	This is type OPToSOAPDomConv.TMultiRefNodeMap.
TMultiRefNodes (see page 2787)	This is type OPToSOAPDomConv.TMultiRefNodes.
TSOAPArrayDesc (see page 2787)	This is type OPToSOAPDomConv.TSOAPArrayDesc.
TUnhandledNodeEvent (see page 2788)	TUnhandledNodeEvent is the type of the OnUnhandledNode event of TSOAPDomConv (see page 2758).
TXMLNodeArray (see page 2788)	This is type OPToSOAPDomConv.TXMLNodeArray.

Variables

Name	Description
DefArrayElemName (see page 2789)	This is variable OPToSOAPDomConv.DefArrayElemName.

51.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	ESOAPDomConvertError (see page 2747)	ESOAPDomConvertError is the class for exceptions that occur when marshaling or unmarshaling a SOAP request.
	TOPToSoapDomConvert (see page 2747)	TOPToSoapDomConvert handles the marshaling and unmarshaling of SOAP method calls.
	TSOAPDomConv (see page 2758)	TSOAPDomConv is the base class for components that use a DOM to interpret and create SOAP method calls.

51.1.1 ESOAPDomConvertError Class

ESOAPDomConvertError is the class for exceptions that occur when marshaling or unmarshaling a SOAP request.

Class Hierarchy

Pascal

```
ESOAPDomConvertError = class(Exception);
```

C++

```
class ESOAPDomConvertError : public Exception;
```

File

OPToSOAPDomConv

Description

TOPToSoapDomConvert (see page 2747) raises ESOAPDomConvertError exceptions when it can't marshal or unmarshal a SOAP packet that encodes a method call or its results.

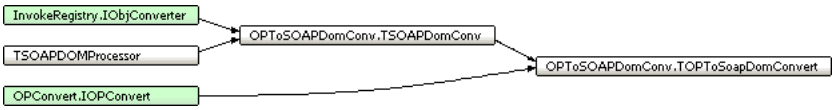
See Also

TOPToSoapDomConvert (see page 2747)

51.1.2 TOPToSoapDomConvert Class

TOPToSoapDomConvert handles the marshaling and unmarshaling of SOAP method calls.

Class Hierarchy



Pascal

```
TOPToSoapDomConvert = class(TSOAPDomConv, IOPConvert);
```

C++

```
class TOPToSoapDomConvert : public TSOAPDomConv, public IOPConvert;
```

File

OPToSOAPDomConv

Description

TOPToSoapDomConvert is an implementation of the IOPConvert (see page 2730) interface that handles the marshaling and unmarshaling of SOAP method calls. It uses a DOM parser to parse and edit the SOAP encoding of a method call and its results.

Typically, applications do not use TOPToSoapDomConvert directly. Rather, It is created and used by THTTPRio on client applications and THTTPSoapPascalInvoker or THTTPSoapCppInvoker on server applications.

TOPToSoapDomConvert converts between the SOAP encoding of a method call or its results and an invocation context. If the method call results in an exception, TOPToSoapDomConvert generates a SOAP error packet for that exception so that the client can raise the exception remotely.

See Also

THTTPRIO

THTTPSoapPascalInvoker

THTTPSoapCppInvoker

IOPConvert (see page 2730)

TInvContext (see page 829)

ESOAPDomConvertError (see page 2747)

51.1.2.1 TOPToSoapDomConvert Members

The following tables list the members exposed by TOPToSoapDomConvert.

Protected Methods

	Name	Description
	AddAttachment (see page 2763)	Stores information about a SOAP attachment so that it is included in the final message that is created.
	AddMultiRefNode (see page 2763)	This is AddMultiRefNode, a member of class TSOAPDomConv.
	AddNamespaceDecl (see page 2763)	This is AddNamespaceDecl, a member of class TSOAPDomConv.
	AddObjectAsWriting (see page 2763)	This is AddObjectAsWriting, a member of class TSOAPDomConv.
	CheckEncodingStyle (see page 2764)	This is CheckEncodingStyle, a member of class TSOAPDomConv.
	ChildNodesAreNull (see page 2764)	This is ChildNodesAreNull, a member of class TSOAPDomConv.
	ConvertByteArrayToSoap (see page 2764)	This is ConvertByteArrayToSoap, a member of class TSOAPDomConv.
	ConvertEnumToSoap (see page 2764)	This is ConvertEnumToSoap, a member of class TSOAPDomConv.
	ConvertNativeArrayToSoap (see page 2765)	This is ConvertNativeArrayToSoap, a member of class TSOAPDomConv.
	ConvertObjectToSOAP (see page 2765)	This is ConvertObjectToSOAP, a member of class TSOAPDomConv.
	ConvertSoapToEnum (see page 2766)	This is ConvertSoapToEnum, a member of class TSOAPDomConv.
	ConvertSoapToNativeArray (see page 2766)	This is ConvertSoapToNativeArray, a member of class TSOAPDomConv.
	ConvertSoapToNativeArrayElem (see page 2766)	This is ConvertSoapToNativeArrayElem, a member of class TSOAPDomConv.
	ConvertSOAPToObject (see page 2767)	This is ConvertSOAPToObject, a member of class TSOAPDomConv.
	ConvertSoapToVariant (see page 2767)	This is ConvertSoapToVariant, a member of class TSOAPDomConv.
	ConvertVariantToSoap (see page 2768)	This is ConvertVariantToSoap, a member of class TSOAPDomConv.

	CreateMultiRefNode (see page 2768)	This is CreateMultiRefNode, a member of class TSOAPDomConv.
	CreateNULLNode (see page 2768)	This is CreateNULLNode, a member of class TSOAPDomConv.
	CreateObjectNode (see page 2768)	This is CreateObjectNode, a member of class TSOAPDomConv.
	CreateScalarNodeXS (see page 2769)	This is CreateScalarNodeXS, a member of class TSOAPDomConv.
	CreateTypedNode (see page 2769)	This is CreateTypedNode, a member of class TSOAPDomConv.
	FinalizeMultiRefNodes (see page 2769)	This is FinalizeMultiRefNodes, a member of class TSOAPDomConv.
	FindAttachment (see page 2769)	Returns a TSOAPAttachment instance that represent the attachment with a specified ID.
	FindMultiRefNodeByID (see page 2770)	This is FindMultiRefNodeByID, a member of class TSOAPDomConv.
	FindMultiRefNodeByInstance (see page 2770)	This is FindMultiRefNodeByInstance, a member of class TSOAPDomConv.
	FindNodeByHREF (see page 2770)	This is FindNodeByHREF, a member of class TSOAPDomConv.
	FindPrefixForURI (see page 2770)	This is FindPrefixForURI, a member of class TSOAPDomConv.
	GetDataNode (see page 2771)	This is GetDataNode, a member of class TSOAPDomConv.
	GetElementType (see page 2771)	This is GetElementType, a member of class TSOAPDomConv.
	GetNewID (see page 2771)	This is GetNewID, a member of class TSOAPDomConv.
	GetNodeAsText (see page 2771)	This is GetNodeAsText, a member of class TSOAPDomConv.
	GetObjectPropAsText (see page 2771)	This is GetObjectPropAsText, a member of class TSOAPDomConv.
	GetOptions (see page 2772)	This is GetOptions, a member of class TSOAPDomConv.
	GetTypeBySchemaNS (see page 2772)	This is GetTypeBySchemaNS, a member of class TSOAPDomConv.
	InitObjectFromSOAP (see page 2772)	Initializes an instance of a remotable object to reflect the values in its SOAP representation.
	IsNodeAVarArray (see page 2773)	This is IsNodeAVarArray, a member of class TSOAPDomConv.
	IsObjectWriting (see page 2773)	This is IsObjectWriting, a member of class TSOAPDomConv.
	LoadObject (see page 2773)	This is LoadObject, a member of class TSOAPDomConv.
	MakeArrayNode (see page 2773)	This is the overview for the MakeArrayNode method overload.
	MultiRefObject (see page 2774)	This is MultiRefObject, a member of class TSOAPDomConv.
	NewXMLDocument (see page 2754)	Creates a new empty XML document to hold the SOAP representation and returns its interface.
	NodeIsNULL (see page 2774)	This is NodeIsNULL, a member of class TSOAPDomConv.
	ObjectMemberNoShow (see page 2775)	This is ObjectMemberNoShow, a member of class TSOAPDomConv.
	ObjInstanceToSOAP (see page 2775)	Returns the SOAP representation of a remotable object.
	ReadHeader (see page 2775)	This is ReadHeader, a member of class TSOAPDomConv.
	ReadRectDynArray (see page 2776)	This is ReadRectDynArray, a member of class TSOAPDomConv.
	ReadRectDynArrayElem (see page 2776)	This is ReadRectDynArrayElem, a member of class TSOAPDomConv.
	ReadRow (see page 2776)	This is ReadRow, a member of class TSOAPDomConv.
	ReadVarArrayDim (see page 2776)	This is ReadVarArrayDim, a member of class TSOAPDomConv.
	ReadVariant (see page 2777)	This is ReadVariant, a member of class TSOAPDomConv.
	RemoveObjectAsWriting (see page 2777)	This is RemoveObjectAsWriting, a member of class TSOAPDomConv.
	ResetMultiRef (see page 2777)	This is ResetMultiRef, a member of class TSOAPDomConv.
	SerializationOptions (see page 2777)	This is the overview for the SerializationOptions method overload.
	SetNodeType (see page 2778)	This is SetNodeType, a member of class TSOAPDomConv.
	SetObjectPropFromText (see page 2778)	This is SetObjectPropFromText, a member of class TSOAPDomConv.
	SetOptions (see page 2779)	This is SetOptions, a member of class TSOAPDomConv.
	UnhandledNode (see page 2779)	This is UnhandledNode, a member of class TSOAPDomConv.
	WriteHeader (see page 2779)	This is WriteHeader, a member of class TSOAPDomConv.
	WriteNonRectDynArray (see page 2779)	This is WriteNonRectDynArray, a member of class TSOAPDomConv.
	WriteNonRectDynArrayElem (see page 2780)	This is WriteNonRectDynArrayElem, a member of class TSOAPDomConv.
	WriteRectDynArray (see page 2780)	This is WriteRectDynArray, a member of class TSOAPDomConv.
	WriteRectDynArrayElem (see page 2780)	This is WriteRectDynArrayElem, a member of class TSOAPDomConv.
	WriteVarArray (see page 2780)	This is WriteVarArray, a member of class TSOAPDomConv.
	WriteVarArrayAsB64 (see page 2781)	This is WriteVarArrayAsB64, a member of class TSOAPDomConv.
	WriteVariant (see page 2781)	This is WriteVariant, a member of class TSOAPDomConv.

Public Methods

	Name	Description
	AddAttachment (see page 921)	Stores information about a SOAP attachment so that it is included in the final message that is created.
	ConvertNativeDataToSoap (see page 2765)	Converts a native type to its SOAP representation.
	ConvertSoapToNativeData (see page 2766)	Converts the SOAP representation of a value to a pointer to the corresponding native representation.
	Create (see page 2751)	Creates a new TOPToSoapDomConvert (see page 2747) object.
	Destroy (see page 2751)	Frees the memory associated with the TOPToSoapDomConvert (see page 2747) object.
	FindAttachment (see page 922)	Returns a TSOAPAttachment (see page 895) instance that represent the attachment with a specified ID.
	GetAttachments (see page 2732)	Returns the value of the Attachments (see page 2738) property.
	GetEncoding (see page 2732)	Returns the value of the Encoding (see page 2738) property.
	GetOptions (see page 2732)	Returns the value of the Options (see page 2739) property.
	GetTempDir (see page 2732)	Returns the value of the TempDir (see page 2739) property.
	InitObjectFromSOAP (see page 922)	Initializes an instance of a remotable object to reflect the values in its SOAP representation.
	InvContextToMsg (see page 2752)	Marshals a method call from an invocation context to a transportable string.
	MakeFault (see page 2752)	Encodes an exception that was raised when trying to execute a method on an invokable interface.
	MakeResponse (see page 2753)	Encodes the results of successfully executing a method on an invokable interface.
	MsgToInvContext (see page 2734)	This is the overview for the MsgToInvContext method overload.
	MsgToInvContext (see page 2753)	This is the overview for the MsgToInvContext method overload.
	ObjInstanceToSOAP (see page 922)	Returns the SOAP representation of a remotable object.
	ProcessResponse (see page 2735)	This is the overview for the ProcessResponse method overload.
	ProcessResponse (see page 2755)	This is the overview for the ProcessResponse method overload.
	SetAttachments (see page 2736)	Sets the value of the Attachments (see page 2738) property.
	SetEncoding (see page 2737)	Sets the value of the Encoding (see page 2738) property.
	SetOptions (see page 2737)	Sets the value of the Options (see page 2739) property.
	SetTempDir (see page 2737)	Sets the value of the TempDir (see page 2739) property.

Public Properties

	Name	Description
	Attachments (see page 2756)	Lists the objects that encapsulate the attachments included with the SOAP request or its response.
	Options (see page 2739)	Controls how certain types of conversions are performed.

Published Events

	Name	Description
	OnMemberDataNotReceived (see page 2782)	Occurs when converting a SOAP representation to a remotable object and an expected class member was not found.
	OnUnhandledNode (see page 2782)	Occurs when a SOAP representation includes an XML node that the converter does not know how to handle.

Published Properties

	Name	Description
	Encoding (see page 2756)	Indicates the character set to use for encoded method calls.
	Options (see page 2781)	Controls how the TSOAPDomConv (see page 2758) object performs certain types of conversions.
	TempDir (see page 2757)	Specifies the directory where temporary files for attachments are saved.
	WSDLView (see page 2758)	Provides access to information in a WSDL document that represents the SOAP request.

51.1.2.2 TOPToSoapDomConvert.Create Constructor

Creates a new TOPToSoapDomConvert (see page 2747) object.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TOPToSoapDomConvert(TComponent * AOwner);
```

Description

Most applications do not create instances of TOPToSoapDomConvert (see page 2747). It is created by the THTTPRio and TSoapPascalInvoker objects to handle the marshaling and unmarshaling of SOAP method calls.

The AOwner parameter specifies the component that is responsible for freeing the TOPToSoapDomConvert (see page 2747) instance.

See Also

Owner

THTTPRIO

TSoapPascalInvoker

51.1.2.3 TOPToSoapDomConvert.Destroy Destructor

Frees the memory associated with the TOPToSoapDomConvert (see page 2747) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TOPToSoapDomConvert();
```

Description

Typically, TOPToSoapDomConvert (see page 2747) objects are freed by the objects that create them to handle the marshaling and unmarshaling of SOAP method calls.

Do not call Destroy directly in an application. Instead, use the Free method, that checks whether the reference is nil before calling Destroy.

See Also

Create (see page 2751)

System::TObject::Free

51.1.2.4 TOPToSoapDomConvert Methods

The methods of the TOPToSoapDomConvert class are listed here.

Protected Methods

	Name	Description
	NewXMLDocument (see page 2754)	Creates a new empty XML document to hold the SOAP representation and returns its interface.

Public Methods

	Name	Description
	InvContextToMsg (see page 2752)	Marshals a method call from an invocation context to a transportable string.
	MakeFault (see page 2752)	Encodes an exception that was raised when trying to execute a method on an invokable interface.
	MakeResponse (see page 2753)	Encodes the results of successfully executing a method on an invokable interface.
	MsgToInvContext (see page 2753)	This is the overview for the MsgToInvContext method overload.
	ProcessResponse (see page 2755)	This is the overview for the ProcessResponse method overload.

51.1.2.4.1 TOPToSoapDomConvert.InvContextToMsg Method

Marshals a method call from an invocation context to a transportable string.

Pascal

```
function InvContextToMsg(const IntfMD: TIntfMetaData; MethNum: Integer; Con: TInvContext;
Headers: THeaderList): TStream;
```

C++

```
__fastcall TStream InvContextToMsg(const TIntfMetaData IntfMD, int MethNum, TInvContext Con,
THeaderList Headers);
```

Description

InvContextToMsg looks up a specified method call, including the types and values of any arguments, from an invocation context, and returns an encoded version of that method call that can be transported to the application that executes it.

IntfMD represents the runtime type information (RTTI) from the interface whose method is called.

MethNum identifies the called method by its position in the interface's method table.

Con is the invocation context that contains details of the method call, including the parameter values passed in, the calling conventions, and any other temporary data used when making the call.

Headers is a list of TSOAPHeader (see page 908) descendants that represent the headers of the SOAP envelope that contained the method call.

InvContextToMsg returns a stream from which an application can read the encoded version of the method call.

See Also

MsgToInvContext (see page 2754)

51.1.2.4.2 TOPToSoapDomConvert.MakeFault Method

Encodes an exception that was raised when trying to execute a method on an invokable interface.

Pascal

```
procedure MakeFault(const Ex: Exception; EStream: TStream); virtual;
```

C++

```
virtual __fastcall MakeFault(const Exception Ex, TStream EStream);
```

Description

MakeFault encodes an exception that a Web Service application catches when trying to execute a method on an invokable interface. It generates an encoded exception packet, which the client can decode and use to raise the exception on the client application.

Ex is the exception object that represents the exception that occurred.

EStream is a stream to which MakeFault writes the encoded exception packet.

See Also

MakeResponse (see page 2753)

51.1.2.4.3 TOPToSoapDomConvert.MakeResponse Method

Encodes the results of successfully executing a method on an invokable interface.

Pascal

```
procedure MakeResponse(const IntfMD: TIntfMetaData; const MethNum: Integer; Context:
TInvContext; Response: TStream; Headers: THeaderLIst); virtual;
```

C++

```
virtual __fastcall MakeResponse(const TIntfMetaData IntfMD, const int MethNum, TInvContext
Context, TStream Response, THeaderLIst Headers);
```

Description

MakeResponse retrieves the results of executing a method on an invokable interface from an invocation context and packages them into a transportable string, which it writes to a stream. This string is passed back to the client, which can unpack it using the ProcessResponse (see page 2755) method.

IntfMD represents the runtime type information (RTTI) from the interface whose method was executed.

MethNum identifies the called method by its position in the interface's method table.

Context is the invocation context that contains details of the method call.

Headers is a list of TSOAPHeader (see page 908) descendants that should be converted to headers on the encoded response.

Response is a stream to which MakeResponse writes the encoded version of the method result and any output parameters.

Headers is a list of TSOAPHeader (see page 908) descendants that represent the headers that should be added to the response message.

See Also

MakeFault (see page 2752)

ProcessResponse (see page 2755)

51.1.2.4.4 MsgToInvContext Method

This is the overview for the MsgToInvContext method overload.

Overload List

	Name	Description
	TOPToSoapDomConvert.MsgToInvContext (InvString, TIntfMetaData, Integer, TInvContext) (see page 2754)	Unpacks an encoded method call into an invocation context so that it can be executed.

	TOPToSoapDomConvert.MsgToInvContext (TStream, TIntfMetaData, Integer, TInvContext, THeaderList) (see page 2754)	Unpacks an encoded method call into an invocation context so that it can be executed.

51.1.2.4.4.1 TOPToSoapDomConvert.MsgToInvContext Method (InvString, TIntfMetaData, Integer, TInvContext)

Unpacks an encoded method call into an invocation context so that it can be executed.

Pascal

```
procedure MsgToInvContext(const Request: InvString; const IntfMD: TIntfMetaData; var MethNum:
Integer; Context: TInvContext); virtual; overload;
procedure MsgToInvContext(const Request: TStream; const IntfMD: TIntfMetaData; var MethNum:
Integer; Context: TInvContext; Headers: THeaderList); virtual; overload;
```

C++

```
virtual __fastcall MsgToInvContext(const InvString Request, const TIntfMetaData IntfMD, int
MethNum, TInvContext Context);
virtual __fastcall MsgToInvContext(const TStream Request, const TIntfMetaData IntfMD, int
MethNum, TInvContext Context, THeaderList Headers);
```

Description

MsgToInvContext takes an encoded message that a server application receives and uses it to fill out an invocation context, which can then be used to execute the method call.

Request is the encoded method call, or a stream from which to read that method call. It is generated on the client by a call to InvContextToMsg (see page 2752).

IntfMD represents the runtime type information (RTTI) from the interface whose method is called.

MethNum returns the position of the called method in the interface's method table.

Context is the invocation context to which this method adds the information encoded in Request.

Headers is an object list to which MsgToInvContext adds TSOAPHeader (see page 908) descendants to represent every header in the encoded message.

See Also

InvContextToMsg (see page 2752)

51.1.2.4.5 TOPToSoapDomConvert.NewXMLDocument Method

Creates a new empty XML document to hold the SOAP representation and returns its interface.

Pascal

```
function NewXMLDocument: IXMLDocument;
```

C++

```
__fastcall IXMLDocument NewXMLDocument();
```

Description

TOPToSoapDomConvert (see page 2747) uses NewXMLDocument internally to create the object that holds the SOAP representation of a message. It returns an interface for working with this representation. Note that the XML document that NewXMLDocument returns is empty. TOPToSoapDomConvert (see page 2747) subsequently uses the methods of the interface to fill out the body of this document.

51.1.2.4.6 ProcessResponse Method

This is the overview for the ProcessResponse method overload.

Overload List

	Name	Description
	TOPToSoapDomConvert.ProcessResponse (InvString, TIntfMetaData, TIntfMethEntry, TInvContext) (see page 2755)	Unpacks the response packet from a method call into an invocation context.
	TOPToSoapDomConvert.ProcessResponse (IXMLDocument, TIntfMetaData, TIntfMethEntry, TInvContext, THeaderList) (see page 2755)	Unpacks the response packet from a method call into an invocation context.
	TOPToSoapDomConvert.ProcessResponse (TStream, TIntfMetaData, TIntfMethEntry, TInvContext, THeaderList) (see page 2755)	Unpacks the response packet from a method call into an invocation context.

51.1.2.4.6.1 TOPToSoapDomConvert.ProcessResponse Method (IXMLDocument, TIntfMetaData, TIntfMethEntry, TInvContext, THeaderList)

Unpacks the response packet from a method call into an invocation context.

Pascal

```
procedure ProcessResponse(const Resp: InvString; const IntfMD: TIntfMetaData; const MD:
TIntfMethEntry; Context: TInvContext); virtual; overload;
procedure ProcessResponse(const Resp: TStream; const IntfMD: TIntfMetaData; const MD:
TIntfMethEntry; Context: TInvContext; Headers: THeaderList); virtual; overload;
procedure ProcessResponse(const XMLDoc: IXMLDocument; const IntfMD: TIntfMetaData; const MD:
TIntfMethEntry; Context: TInvContext; Headers: THeaderList); virtual; overload;
```

C++

```
virtual __fastcall ProcessResponse(const InvString Resp, const TIntfMetaData IntfMD, const
TIntfMethEntry MD, TInvContext Context);
virtual __fastcall ProcessResponse(const TStream Resp, const TIntfMetaData IntfMD, const
TIntfMethEntry MD, TInvContext Context, THeaderList Headers);
virtual __fastcall ProcessResponse(const IXMLDocument XMLDoc, const TIntfMetaData IntfMD,
const TIntfMethEntry MD, TInvContext Context, THeaderList Headers);
```

Description

ProcessResponse takes the encoded results of executing a method on an invokable interface from an invocation context and unpacks them into an invocation context.

Response is the encoded results of executing a method on the invokable interface, or a stream from which those results can be read. It is generated on the server by the MakeResponse (see page 2753) method if the call was successful, or the MakeFault (see page 2752) method if the call was unsuccessful.

XMLDoc is the interface to an XML document that represents the results of executing a method on the invokable interface. The values of its nodes come from the MakeResponse (see page 2753) method if the call was successful, or the MakeFault (see page 2752) method if the call was unsuccessful.

IntfMD describes the runtime type information (RTTI) of the interface whose method was called.

MD describes the runtime type information (RTTI) of the method that was called.

Context is the invocation context that ProcessResponse fills out based on the contents of Response.

Headers is a list that ProcessResponse fills with TSOAPHeader (see page 908) descendants that represent the headers of the

encoded result message.

See Also

MakeFault (see page 2752)

MakeResponse (see page 2753)

51.1.2.5 TOPToSoapDomConvert Properties

The properties of the TOPToSoapDomConvert class are listed here.

Public Properties

	Name	Description
	Attachments (see page 2756)	Lists the objects that encapsulate the attachments included with the SOAP request or its response.

Published Properties

	Name	Description
	Encoding (see page 2756)	Indicates the character set to use for encoded method calls.
	TempDir (see page 2757)	Specifies the directory where temporary files for attachments are saved.
	WSDLView (see page 2758)	Provides access to information in a WSDL document that represents the SOAP request.

51.1.2.5.1 TOPToSoapDomConvert.Attachments Property

Lists the objects that encapsulate the attachments included with the SOAP request or its response.

Pascal

```
property Attachments: TSOAPDataList;
```

C++

```
__property TSOAPDataList Attachments;
```

Description

When a method on an invokable interface requires the user to send or receive attachments, these attachments are each represented by a TSoapAttachment object. Attachments lists the TSoapAttachment objects for each attachment of the current method.

Note: When the application that contains TOPToSoapDomConvert (see page 2747) receives attachments, it writes them to a temporary file rather than storing them in memory. The location for these temporary files is specified by the TempDir (see page 2757) property.

See Also

TempDir (see page 2757)

TSOAPAttachment (see page 895)

51.1.2.5.2 TOPToSoapDomConvert.Encoding Property

Indicates the character set to use for encoded method calls.

Pascal

```
property Encoding: DOMString;
```

C++

```
__property DOMString Encoding;
```

Description

Use Encoding to indicate the character set that is used for encoded method calls. Encoding causes the converter to make character set conversions using the DOM, rather than at the transport level. This approach is more efficient than transport-level encoding. If you set Encoding, the Options (see page 2781) property should not include soUTF8EncodeXML and soUTF8InHeader.

If you do not specify an encoding (that is, if Encoding is an empty string), TOPToSoapDomConvert (see page 2747) converts characters at the transport level assuming UTF-8.

TOPToSoapDomConvert (see page 2747) uses the DOM parser identified by the global DefaultDOMVendor variable to perform its conversions. Encoding must indicate an encoding that this default DOM parser supports.

Note: Some Web Services (such as Apache or .NET) require transport level encoding rather than DOM level encoding. In such cases, it is possible that setting Encoding can cause problems because the Web Service converts characters at the transport level and then converts a second time because of the encoding attribute that Encoding adds to the XML representation. If a Web Service requires transport-level encoding and assumes UTF8, Encoding should be an empty string and the Options (see page 2781) property should include soUTF8EncodeXML and soUTF8InHeader.

See Also

DefaultDOMVendor

Encoding

Options (see page 2781)

51.1.2.5.3 TOPToSoapDomConvert.TempDir Property

Specifies the directory where temporary files for attachments are saved.

Pascal

```
property TempDir: string;
```

C++

```
__property UnicodeString TempDir;
```

Description

When the application that includes this TOPToSoapDomConvert (see page 2747) instance receives an attachment as the value of a parameter (or the return value) of the current method, TOPToSoapDomConvert (see page 2747) writes the attachment to a temporary file. This temporary file becomes the value of the CacheFile property on the TSoapAttachment object that represents the attachment.

Use TempDir to specify the directory where TOPToSoapDomConvert (see page 2747) writes the temporary files for attachments.

Note: If TempDir is not set, the application uses the default temporary directory. On Windows, this is the directory that the Windows API GetTempPath returns. On Linux, this is obtained from the TMPDIR environment setting (or, if TMPDIR is not set, /tmp/)

See Also

Attachments (see page 2756)

CacheFile (see page 904)

51.1.2.5.4 TOPToSoapDomConvert.WSDLView Property

Provides access to information in a WSDL document that represents the SOAP request.

Pascal

```
property WSDLView: TWSDLView;
```

C++

```
__property TWSDLView WSDLView;
```

Description

WSDLView represents information about the SOAP request when the application uses a WSDL document to define information about the interface. If the application does not use a WSDL document, WSDLView is nil (Delphi) or NULL (C++).

WSDL includes the parsed WSDL document, as well as properties to identify the target method call.

51.1.3 TSOAPDomConv Class

TSOAPDomConv is the base class for components that use a DOM to interpret and create SOAP method calls.

Class Hierarchy

Pascal

```
TSOAPDomConv = class(TSOAPDOMProcessor, IObjConverter);
```

C++

```
class TSOAPDomConv : public TSOAPDOMProcessor, public IObjConverter;
```

File

OPToSOAPDomConv

Description

TSOAPDomConv is the base class for DOM-based components that interpret and create SOAP method calls. It handles the logic of converting between the SOAP representation of a type and its native representation. TSOAPDomConv uses the remotable type registry (RemTypeRegistry (see page 931)) to look up the native type or class that corresponds to a type from a SOAP method call and to generate an instance of that type to represent an incoming value. It uses runtime type information (RTTI) to convert a native type or remotable class to its SOAP representation.

TSOAPDomConv implements the IObjConverter (see page 920) interface.

Do not create instances of TSOAPDomConv. It is intended only as a base class for components that interpret or create SOAP method calls.

See Also

TOPToSoapDomConvert (see page 2747)

RemTypeRegistry (see page 931)

IObjConverter (see page 920)

51.1.3.1 TSOAPDomConv Members

The following tables list the members exposed by TSOAPDomConv.

Protected Methods

	Name	Description
	AddAttachment (see page 2763)	Stores information about a SOAP attachment so that it is included in the final message that is created.
	AddMultiRefNode (see page 2763)	This is AddMultiRefNode, a member of class TSOAPDomConv.
	AddNamespaceDecl (see page 2763)	This is AddNamespaceDecl, a member of class TSOAPDomConv.
	AddObjectAsWriting (see page 2763)	This is AddObjectAsWriting, a member of class TSOAPDomConv.
	CheckEncodingStyle (see page 2764)	This is CheckEncodingStyle, a member of class TSOAPDomConv.
	ChildNodesAreNull (see page 2764)	This is ChildNodesAreNull, a member of class TSOAPDomConv.
	ConvertByteArrayToSoap (see page 2764)	This is ConvertByteArrayToSoap, a member of class TSOAPDomConv.
	ConvertEnumToSoap (see page 2764)	This is ConvertEnumToSoap, a member of class TSOAPDomConv.
	ConvertNativeArrayToSoap (see page 2765)	This is ConvertNativeArrayToSoap, a member of class TSOAPDomConv.
	ConvertObjectToSOAP (see page 2765)	This is ConvertObjectToSOAP, a member of class TSOAPDomConv.
	ConvertSoapToEnum (see page 2766)	This is ConvertSoapToEnum, a member of class TSOAPDomConv.
	ConvertSoapToNativeArray (see page 2766)	This is ConvertSoapToNativeArray, a member of class TSOAPDomConv.
	ConvertSoapToNativeArrayElem (see page 2766)	This is ConvertSoapToNativeArrayElem, a member of class TSOAPDomConv.
	ConvertSOAPToObject (see page 2767)	This is ConvertSOAPToObject, a member of class TSOAPDomConv.
	ConvertSoapToVariant (see page 2767)	This is ConvertSoapToVariant, a member of class TSOAPDomConv.
	ConvertVariantToSoap (see page 2768)	This is ConvertVariantToSoap, a member of class TSOAPDomConv.
	CreateMultiRefNode (see page 2768)	This is CreateMultiRefNode, a member of class TSOAPDomConv.
	CreateNULLNode (see page 2768)	This is CreateNULLNode, a member of class TSOAPDomConv.
	CreateObjectNode (see page 2768)	This is CreateObjectNode, a member of class TSOAPDomConv.
	CreateScalarNodeXS (see page 2769)	This is CreateScalarNodeXS, a member of class TSOAPDomConv.
	CreateTypedNode (see page 2769)	This is CreateTypedNode, a member of class TSOAPDomConv.
	FinalizeMultiRefNodes (see page 2769)	This is FinalizeMultiRefNodes, a member of class TSOAPDomConv.
	FindAttachment (see page 2769)	Returns a TSOAPAttachment instance that represent the attachment with a specified ID.
	FindMultiRefNodeByID (see page 2770)	This is FindMultiRefNodeByID, a member of class TSOAPDomConv.
	FindMultiRefNodeByInstance (see page 2770)	This is FindMultiRefNodeByInstance, a member of class TSOAPDomConv.
	FindNodeByHREF (see page 2770)	This is FindNodeByHREF, a member of class TSOAPDomConv.
	FindPrefixForURI (see page 2770)	This is FindPrefixForURI, a member of class TSOAPDomConv.
	GetDataNode (see page 2771)	This is GetDataNode, a member of class TSOAPDomConv.
	GetElementType (see page 2771)	This is GetElementType, a member of class TSOAPDomConv.
	GetNewID (see page 2771)	This is GetNewID, a member of class TSOAPDomConv.
	GetNodeAsText (see page 2771)	This is GetNodeAsText, a member of class TSOAPDomConv.
	GetObjectPropAsText (see page 2771)	This is GetObjectPropAsText, a member of class TSOAPDomConv.
	GetOptions (see page 2772)	This is GetOptions, a member of class TSOAPDomConv.
	GetTypeBySchemaNS (see page 2772)	This is GetTypeBySchemaNS, a member of class TSOAPDomConv.
	InitObjectFromSOAP (see page 2772)	Initializes an instance of a remotable object to reflect the values in its SOAP representation.
	IsNodeAVarArray (see page 2773)	This is IsNodeAVarArray, a member of class TSOAPDomConv.
	IsObjectWriting (see page 2773)	This is IsObjectWriting, a member of class TSOAPDomConv.
	LoadObject (see page 2773)	This is LoadObject, a member of class TSOAPDomConv.
	MakeArrayNode (see page 2773)	This is the overview for the MakeArrayNode method overload.
	MultiRefObject (see page 2774)	This is MultiRefObject, a member of class TSOAPDomConv.
	NodeIsNULL (see page 2774)	This is NodeIsNULL, a member of class TSOAPDomConv.
	ObjectMemberNoShow (see page 2775)	This is ObjectMemberNoShow, a member of class TSOAPDomConv.
	ObjInstanceToSOAP (see page 2775)	Returns the SOAP representation of a remotable object.
	ReadHeader (see page 2775)	This is ReadHeader, a member of class TSOAPDomConv.
	ReadRectDynArray (see page 2776)	This is ReadRectDynArray, a member of class TSOAPDomConv.

	ReadRectDynArrayElem (see page 2776)	This is ReadRectDynArrayElem, a member of class TSOAPDomConv.
	ReadRow (see page 2776)	This is ReadRow, a member of class TSOAPDomConv.
	ReadVarArrayDim (see page 2776)	This is ReadVarArrayDim, a member of class TSOAPDomConv.
	ReadVariant (see page 2777)	This is ReadVariant, a member of class TSOAPDomConv.
	RemoveObjectAsWriting (see page 2777)	This is RemoveObjectAsWriting, a member of class TSOAPDomConv.
	ResetMultiRef (see page 2777)	This is ResetMultiRef, a member of class TSOAPDomConv.
	SerializationOptions (see page 2777)	This is the overview for the SerializationOptions method overload.
	SetNodeType (see page 2778)	This is SetNodeType, a member of class TSOAPDomConv.
	SetObjectPropFromText (see page 2778)	This is SetObjectPropFromText, a member of class TSOAPDomConv.
	SetOptions (see page 2779)	This is SetOptions, a member of class TSOAPDomConv.
	UnhandledNode (see page 2779)	This is UnhandledNode, a member of class TSOAPDomConv.
	WriteHeader (see page 2779)	This is WriteHeader, a member of class TSOAPDomConv.
	WriteNonRectDynArray (see page 2779)	This is WriteNonRectDynArray, a member of class TSOAPDomConv.
	WriteNonRectDynArrayElem (see page 2780)	This is WriteNonRectDynArrayElem, a member of class TSOAPDomConv.
	WriteRectDynArray (see page 2780)	This is WriteRectDynArray, a member of class TSOAPDomConv.
	WriteRectDynArrayElem (see page 2780)	This is WriteRectDynArrayElem, a member of class TSOAPDomConv.
	WriteVarArray (see page 2780)	This is WriteVarArray, a member of class TSOAPDomConv.
	WriteVarArrayAsB64 (see page 2781)	This is WriteVarArrayAsB64, a member of class TSOAPDomConv.
	WriteVariant (see page 2781)	This is WriteVariant, a member of class TSOAPDomConv.

Public Methods

	Name	Description
	ConvertNativeDataToSoap (see page 2765)	Converts a native type to its SOAP representation.
	ConvertSoapToNativeData (see page 2766)	Converts the SOAP representation of a value to a pointer to the corresponding native representation.
V	Create (see page 2760)	Creates a new TSOAPDomConv (see page 2758) object.
V	Destroy (see page 2761)	Frees the memory associated with the TSOAPDomConv (see page 2758) object.

Published Events

	Name	Description
	OnMemberDataNotReceived (see page 2782)	Occurs when converting a SOAP representation to a remotable object and an expected class member was not found.
	OnUnhandledNode (see page 2782)	Occurs when a SOAP representation includes an XML node that the converter does not know how to handle.

Published Properties

	Name	Description
	Options (see page 2781)	Controls how the TSOAPDomConv (see page 2758) object performs certain types of conversions.

51.1.3.2 TSOAPDomConv.Create Constructor

Creates a new TSOAPDomConv (see page 2758) object.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TSOAPDomConv(TComponent * AOwner);
```

Description

Typically, applications do not need to create instances of TSOAPDomConv (see page 2758). Instead, it is created and used by THTTPRio on client applications and THTTPSoapPascalInvoker or THTTPSoapCppInvoker on server applications.

The AOwner parameter specifies the component that is responsible for freeing the TSOAPDomConv (see page 2758) instance.

See Also

Owner

THTTPRIO

THTTPSoapPascalInvoker

THTTPSoapCppInvoker

51.1.3.3 TSOAPDomConv.Destroy Destructor

Frees the memory associated with the TSOAPDomConv (see page 2758) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TSOAPDomConv();
```

Description

Do not call Destroy to free an instance of TSOAPDomConv (see page 2758). Instead, use the Free method, which first checks for a nil reference before calling Destroy.

See Also

System::TObject::Free

51.1.3.4 TSOAPDomConv Methods

The methods of the TSOAPDomConv class are listed here.

Protected Methods

	Name	Description
	AddAttachment (see page 2763)	Stores information about a SOAP attachment so that it is included in the final message that is created.
	AddMultiRefNode (see page 2763)	This is AddMultiRefNode, a member of class TSOAPDomConv.
	AddNamespaceDecl (see page 2763)	This is AddNamespaceDecl, a member of class TSOAPDomConv.
	AddObjectAsWriting (see page 2763)	This is AddObjectAsWriting, a member of class TSOAPDomConv.
	CheckEncodingStyle (see page 2764)	This is CheckEncodingStyle, a member of class TSOAPDomConv.
	ChildNodesAreNull (see page 2764)	This is ChildNodesAreNull, a member of class TSOAPDomConv.
	ConvertByteArrayToSoap (see page 2764)	This is ConvertByteArrayToSoap, a member of class TSOAPDomConv.
	ConvertEnumToSoap (see page 2764)	This is ConvertEnumToSoap, a member of class TSOAPDomConv.
	ConvertNativeArrayToSoap (see page 2765)	This is ConvertNativeArrayToSoap, a member of class TSOAPDomConv.
	ConvertObjectToSOAP (see page 2765)	This is ConvertObjectToSOAP, a member of class TSOAPDomConv.
	ConvertSoapToEnum (see page 2766)	This is ConvertSoapToEnum, a member of class TSOAPDomConv.
	ConvertSoapToNativeArray (see page 2766)	This is ConvertSoapToNativeArray, a member of class TSOAPDomConv.
	ConvertSoapToNativeArrayElem (see page 2766)	This is ConvertSoapToNativeArrayElem, a member of class TSOAPDomConv.
	ConvertSOAPToObject (see page 2767)	This is ConvertSOAPToObject, a member of class TSOAPDomConv.
	ConvertSoapToVariant (see page 2767)	This is ConvertSoapToVariant, a member of class TSOAPDomConv.
	ConvertVariantToSoap (see page 2768)	This is ConvertVariantToSoap, a member of class TSOAPDomConv.
	CreateMultiRefNode (see page 2768)	This is CreateMultiRefNode, a member of class TSOAPDomConv.
	CreateNULLNode (see page 2768)	This is CreateNULLNode, a member of class TSOAPDomConv.
	CreateObjectNode (see page 2768)	This is CreateObjectNode, a member of class TSOAPDomConv.
	CreateScalarNodeXS (see page 2769)	This is CreateScalarNodeXS, a member of class TSOAPDomConv.

	CreateTypedNode (see page 2769)	This is CreateTypedNode, a member of class TSOAPDomConv.
	FinalizeMultiRefNodes (see page 2769)	This is FinalizeMultiRefNodes, a member of class TSOAPDomConv.
	FindAttachment (see page 2769)	Returns a TSOAPAttachment instance that represent the attachment with a specified ID.
	FindMultiRefNodeByID (see page 2770)	This is FindMultiRefNodeByID, a member of class TSOAPDomConv.
	FindMultiRefNodeByInstance (see page 2770)	This is FindMultiRefNodeByInstance, a member of class TSOAPDomConv.
	FindNodeByHREF (see page 2770)	This is FindNodeByHREF, a member of class TSOAPDomConv.
	FindPrefixForURI (see page 2770)	This is FindPrefixForURI, a member of class TSOAPDomConv.
	GetDataNode (see page 2771)	This is GetDataNode, a member of class TSOAPDomConv.
	GetElementType (see page 2771)	This is GetElementType, a member of class TSOAPDomConv.
	GetNewID (see page 2771)	This is GetNewID, a member of class TSOAPDomConv.
	GetNodeAsText (see page 2771)	This is GetNodeAsText, a member of class TSOAPDomConv.
	GetObjectPropAsText (see page 2771)	This is GetObjectPropAsText, a member of class TSOAPDomConv.
	GetOptions (see page 2772)	This is GetOptions, a member of class TSOAPDomConv.
	GetTypeBySchemaNS (see page 2772)	This is GetTypeBySchemaNS, a member of class TSOAPDomConv.
	InitObjectFromSOAP (see page 2772)	Initializes an instance of a remotable object to reflect the values in its SOAP representation.
	IsNodeAVarArray (see page 2773)	This is IsNodeAVarArray, a member of class TSOAPDomConv.
	IsObjectWriting (see page 2773)	This is IsObjectWriting, a member of class TSOAPDomConv.
	LoadObject (see page 2773)	This is LoadObject, a member of class TSOAPDomConv.
	MakeArrayNode (see page 2773)	This is the overview for the MakeArrayNode method overload.
	MultiRefObject (see page 2774)	This is MultiRefObject, a member of class TSOAPDomConv.
	NodeIsNULL (see page 2774)	This is NodeIsNULL, a member of class TSOAPDomConv.
	ObjectMemberNoShow (see page 2775)	This is ObjectMemberNoShow, a member of class TSOAPDomConv.
	ObjInstanceToSOAP (see page 2775)	Returns the SOAP representation of a remotable object.
	ReadHeader (see page 2775)	This is ReadHeader, a member of class TSOAPDomConv.
	ReadRectDynArray (see page 2776)	This is ReadRectDynArray, a member of class TSOAPDomConv.
	ReadRectDynArrayElem (see page 2776)	This is ReadRectDynArrayElem, a member of class TSOAPDomConv.
	ReadRow (see page 2776)	This is ReadRow, a member of class TSOAPDomConv.
	ReadVarArrayDim (see page 2776)	This is ReadVarArrayDim, a member of class TSOAPDomConv.
	ReadVariant (see page 2777)	This is ReadVariant, a member of class TSOAPDomConv.
	RemoveObjectAsWriting (see page 2777)	This is RemoveObjectAsWriting, a member of class TSOAPDomConv.
	ResetMultiRef (see page 2777)	This is ResetMultiRef, a member of class TSOAPDomConv.
	SerializationOptions (see page 2777)	This is the overview for the SerializationOptions method overload.
	SetNodeType (see page 2778)	This is SetNodeType, a member of class TSOAPDomConv.
	SetObjectPropFromText (see page 2778)	This is SetObjectPropFromText, a member of class TSOAPDomConv.
	SetOptions (see page 2779)	This is SetOptions, a member of class TSOAPDomConv.
	UnhandledNode (see page 2779)	This is UnhandledNode, a member of class TSOAPDomConv.
	WriteHeader (see page 2779)	This is WriteHeader, a member of class TSOAPDomConv.
	WriteNonRectDynArray (see page 2779)	This is WriteNonRectDynArray, a member of class TSOAPDomConv.
	WriteNonRectDynArrayElem (see page 2780)	This is WriteNonRectDynArrayElem, a member of class TSOAPDomConv.
	WriteRectDynArray (see page 2780)	This is WriteRectDynArray, a member of class TSOAPDomConv.
	WriteRectDynArrayElem (see page 2780)	This is WriteRectDynArrayElem, a member of class TSOAPDomConv.
	WriteVarArray (see page 2780)	This is WriteVarArray, a member of class TSOAPDomConv.
	WriteVarArrayAsB64 (see page 2781)	This is WriteVarArrayAsB64, a member of class TSOAPDomConv.
	WriteVariant (see page 2781)	This is WriteVariant, a member of class TSOAPDomConv.

Public Methods

	Name	Description
	ConvertNativeDataToSoap (see page 2765)	Converts a native type to its SOAP representation.
	ConvertSoapToNativeData (see page 2766)	Converts the SOAP representation of a value to a pointer to the corresponding native representation.

51.1.3.4.1 TSOAPDomConv.AddAttachment Method

Stores information about a SOAP attachment so that it is included in the final message that is created.

Pascal

```
procedure AddAttachment(Attachment: TSOAPAttachment; const AContentId: string);
```

C++

```
__fastcall AddAttachment(TSOAPAttachment Attachment, const UnicodeString AContentId);
```

Description

AddAttachment stores a SOAP attachment so that it can be used to generate part of the multipart form that includes the SOAP envelope as another of its parts.

Attachment is the attachment that should be added to the multipart form.

AContentId is a string that uniquely identifies the attachment. It is added as a header to the part of the form that contains the attachment so that XML in the SOAP envelope can refer to the attachment. When interpreting the encoded SOAP envelope, this value is passed to the FindAttachment method so that the information in the attachment part can be used to recreate the TSOAPAttachment instance.

See Also

FindAttachment (see page 922)

51.1.3.4.2 TSOAPDomConv.AddMultiRefNode Method

Pascal

```
procedure AddMultiRefNode(const ID: string; Instance: Pointer);
```

C++

```
__fastcall AddMultiRefNode(const UnicodeString ID, void * Instance);
```

Description

This is AddMultiRefNode, a member of class TSOAPDomConv.

51.1.3.4.3 TSOAPDomConv.AddNamespaceDecl Method

Pascal

```
function AddNamespaceDecl(Node: IXMLNode; const URI: InvString): InvString;
```

C++

```
__fastcall InvString AddNamespaceDecl(IXMLNode Node, const InvString URI);
```

Description

This is AddNamespaceDecl, a member of class TSOAPDomConv.

51.1.3.4.4 TSOAPDomConv.AddObjectAsWriting Method

Pascal

```
procedure AddObjectAsWriting(Instance: TObject);
```

C++

```
__fastcall AddObjectAsWriting(TObject * Instance);
```

Description

This is AddObjectAsWriting, a member of class TSOAPDomConv.

51.1.3.4.5 TSOAPDomConv.CheckEncodingStyle Method

Pascal

```
procedure CheckEncodingStyle(Node: IXMLNode);
```

C++

```
__fastcall CheckEncodingStyle(IXMLNode Node);
```

Description

This is CheckEncodingStyle, a member of class TSOAPDomConv.

51.1.3.4.6 TSOAPDomConv.ChildNodesAreNull Method

Pascal

```
function ChildNodesAreNull(Node: IXMLNode): Boolean;
```

C++

```
__fastcall Boolean ChildNodesAreNull(IXMLNode Node);
```

Description

This is ChildNodesAreNull, a member of class TSOAPDomConv.

51.1.3.4.7 TSOAPDomConv.ConvertByteArrayToSoap Method

Pascal

```
procedure ConvertByteArrayToSoap(RootNode: IXMLNode; Node: IXMLNode; const Name: InvString;
Info: PTypeInfo; P: Pointer);
```

C++

```
__fastcall ConvertByteArrayToSoap(IXMLNode RootNode, IXMLNode Node, const InvString Name,
PTypeInfo Info, void * P);
```

Description

This is ConvertByteArrayToSoap, a member of class TSOAPDomConv.

51.1.3.4.8 TSOAPDomConv.ConvertEnumToSoap Method

Pascal

```
function ConvertEnumToSoap(Info: PTypeInfo; P: Pointer; NumIndirect: Integer): InvString;
```

C++

```
__fastcall InvString ConvertEnumToSoap(PTypeInfo Info, void * P, int NumIndirect);
```

Description

This is ConvertEnumToSoap, a member of class TSOAPDomConv.

51.1.3.4.9 TSOAPDomConv.ConvertNativeArrayToSoap Method

Pascal

```
procedure ConvertNativeArrayToSoap(RootNode: IXMLNode; Node: IXMLNode; const Name: InvString;
Info: PTypeInfo; P: Pointer; NumIndirect: Integer; InlineElements: Boolean = False);
```

C++

```
__fastcall ConvertNativeArrayToSoap(IXMLNode RootNode, IXMLNode Node, const InvString Name,
PTypeInfo Info, void * P, int NumIndirect, Boolean InlineElements = False);
```

Description

This is ConvertNativeArrayToSoap, a member of class TSOAPDomConv.

51.1.3.4.10 TSOAPDomConv.ConvertNativeDataToSoap Method

Converts a native type to its SOAP representation.

Pascal

```
procedure ConvertNativeDataToSoap(RootNode: IXMLNode; Node: IXMLNode; const Name: InvString;
Info: PTypeInfo; P: Pointer; NumIndirect: Integer); dynamic;
```

C++

```
__fastcall ConvertNativeDataToSoap(IXMLNode RootNode, IXMLNode Node, const InvString Name,
PTypeInfo Info, void * P, int NumIndirect);
```

Description

ConvertNativeDataToSoap performs the conversion from a native type to its SOAP representation and adds that to the emerging DOM representation of a SOAP packet. It uses the options specified by the Options (see page 2781) property to determine whether to include type data and how to handle multiple references to the same remotable object.

RootNode is the node in the emerging SOAP representation that acts as a parent to any elements that represent an object that has multiple references.

Node is the parent node to which ConvertNativeDataToSoap adds a new child node that represents the native value.

Name is the type name of the SOAP encoding of the native type.

Info is a TypeInfo pointer that indicates the type in which the native data appears.

P is a pointer to the value to convert. It points (possibly with additional levels of indirection) to a value of the type indicated by Info.

NumIndirect indicates the number of levels of indirection by which P points to a value of the type indicated by Info.

See Also

ConvertSoapToNativeData (see page 2766)

Options (see page 2781)

51.1.3.4.11 TSOAPDomConv.ConvertObjectToSOAP Method

Pascal

```
procedure ConvertObjectToSOAP(const Name: InvString; ObjP: Pointer; RootNode: IXMLNOde; Node:
IXMLNOde; NumIndirect: Integer);
```

C++

```
__fastcall ConvertObjectToSOAP(const InvString Name, void * ObjP, IXMLNOde RootNode, IXMLNOde
Node, int NumIndirect);
```

Description

This is ConvertObjectToSOAP, a member of class TSOAPDomConv.

51.1.3.4.12 TSOAPDomConv.ConvertSoapToEnum Method

Pascal

```
function ConvertSoapToEnum(Info: PTypeInfo; S: InvString; IsNull: Boolean): Integer;
```

C++

```
__fastcall int ConvertSoapToEnum(PTypeInfo Info, InvString S, Boolean IsNull);
```

Description

This is ConvertSoapToEnum, a member of class TSOAPDomConv.

51.1.3.4.13 TSOAPDomConv.ConvertSoapToNativeArray Method

Pascal

```
function ConvertSoapToNativeArray(DataP: Pointer; TypeInfo: PTypeInfo; RootNode: IXMLNode;
Node: IXMLNode): Pointer;
```

C++

```
__fastcall void * ConvertSoapToNativeArray(void * DataP, PTypeInfo TypeInfo, IXMLNode
RootNode, IXMLNode Node);
```

Description

This is ConvertSoapToNativeArray, a member of class TSOAPDomConv.

51.1.3.4.14 TSOAPDomConv.ConvertSoapToNativeArrayElem Method

Pascal

```
function ConvertSoapToNativeArrayElem(ArrayInfo: PTypeInfo; ElemInfo: PTypeInfo; RootNode:
IXMLNode; Node: IXMLNode; ArrayDesc: TSOAPArrayDesc; Dims: Integer; CurDim: Integer; DataP:
Pointer): Pointer;
```

C++

```
__fastcall void * ConvertSoapToNativeArrayElem(PTypeInfo ArrayInfo, PTypeInfo ElemInfo,
IXMLNode RootNode, IXMLNode Node, TSOAPArrayDesc ArrayDesc, int Dims, int CurDim, void *
DataP);
```

Description

This is ConvertSoapToNativeArrayElem, a member of class TSOAPDomConv.

51.1.3.4.15 TSOAPDomConv.ConvertSoapToNativeData Method

Converts the SOAP representation of a value to a pointer to the corresponding native representation.

Pascal

```
procedure ConvertSoapToNativeData(DataP: Pointer; TypeInfo: PTypeInfo; Context: TDataContext;
RootNode: IXMLNode; Node: IXMLNode; Translate: Boolean; ByRef: Boolean; NumIndirect: Integer);
```

```
dynamic;
```

C++

```
__fastcall ConvertSoapToNativeData(void * DataP, PTypeInfo TypeInfo, TDataContext Context,
IXMLNode RootNode, IXMLNode Node, Boolean Translate, Boolean ByRef, int NumIndirect);
```

Description

ConvertSoapToNativeData performs the conversion from a DOM node that represents a SOAP encoding of a value to the corresponding native representation.

DataP is a pointer to the memory allocated to hold the native representation of the value, and points to the results of the conversion.

TypeInfo is the typeinfo pointer that describes the native type that will represent the value.

Context is the TDataContext (see page 814) object that stores the value.

RootNode is the node in the SOAP packet that acts as a parent to any elements to which there are multiple references.

Node is the node in the SOAP packet whose value is to be translated.

Translate indicates whether Node represents a value to be translated, as opposed to a reference to value that is found elsewhere in the SOAP packet.

ByRef indicates whether the node corresponds to a parameter that is passed by reference.

NumIndirect indicates the number of levels of indirection by which DataP points to its value.

See Also

ConvertNativeDataToSoap (see page 2765)

Options (see page 2781)

51.1.3.4.16 TSOAPDomConv.ConvertSOAPToObject Method

Pascal

```
function ConvertSOAPToObject(RootNode: IXMLNode; Node: IXMLNode; AClass: TClass; const URI:
InvString; const TypeName: InvString; ObjP: Pointer; NumIndirect: Integer): TObject;
```

C++

```
__fastcall TObject * ConvertSOAPToObject(IXMLNode RootNode, IXMLNode Node, TClass AClass,
const InvString URI, const InvString TypeName, void * ObjP, int NumIndirect);
```

Description

This is ConvertSOAPToObject, a member of class TSOAPDomConv.

51.1.3.4.17 TSOAPDomConv.ConvertSoapToVariant Method

Pascal

```
procedure ConvertSoapToVariant(Node: IXMLNode; InvData: Pointer);
```

C++

```
__fastcall ConvertSoapToVariant(IXMLNode Node, void * InvData);
```

Description

This is ConvertSoapToVariant, a member of class TSOAPDomConv.

51.1.3.4.18 TSOAPDomConv.ConvertVariantToSoap Method

Pascal

```
procedure ConvertVariantToSoap(RootNode: IXMLNode; Node: IXMLNode; const Name: InvString;
Info: PTypeInfo; P: PVarData; NumIndirect: Integer; V: Variant; UseVariant: Boolean);
```

C++

```
__fastcall ConvertVariantToSoap(IXMLNode RootNode, IXMLNode Node, const InvString Name,
PTypeInfo Info, PVarData P, int NumIndirect, Variant V, Boolean UseVariant);
```

Description

This is ConvertVariantToSoap, a member of class TSOAPDomConv.

51.1.3.4.19 TSOAPDomConv.CreateMultiRefNode Method

Pascal

```
function CreateMultiRefNode(RootNode: IXMLNode; const Name: InvString; const ID: InvString):
IXMLNode;
```

C++

```
__fastcall IXMLNode CreateMultiRefNode(IXMLNode RootNode, const InvString Name, const
InvString ID);
```

Description

This is CreateMultiRefNode, a member of class TSOAPDomConv.

51.1.3.4.20 TSOAPDomConv.CreateNULLNode Method

Pascal

```
function CreateNULLNode(RootNode: IXMLNode; ParentNode: IXMLNode; const Name: InvString;
UseParentNode: Boolean = False): IXMLNode;
```

C++

```
__fastcall IXMLNode CreateNULLNode(IXMLNode RootNode, IXMLNode ParentNode, const InvString
Name, Boolean UseParentNode = False);
```

Description

This is CreateNULLNode, a member of class TSOAPDomConv.

51.1.3.4.21 TSOAPDomConv.CreateObjectNode Method

Pascal

```
function CreateObjectNode(Instance: TObject; RootNode: IXMLNode; ParentNode: IXMLNode; const
Name: InvString; const URI: InvString; ObjConvOpts: TObjectConvertOptions): InvString;
```

C++

```
__fastcall InvString CreateObjectNode(TObject * Instance, IXMLNode RootNode, IXMLNode
ParentNode, const InvString Name, const InvString URI, TObjectConvertOptions ObjConvOpts);
```

Description

This is CreateObjectNode, a member of class TSOAPDomConv.

51.1.3.4.22 TSOAPDomConv.CreateScalarNodeXS Method

Pascal

```
function CreateScalarNodeXS(RootNode: IXMLNode; ParentNode: IXMLNode; const NodeName:
InvString; const URI: InvString; const TypeName: InvString; const Value: InvString; GenPre:
Boolean = False): IXMLNode;
```

C++

```
__fastcall IXMLNode CreateScalarNodeXS(IXMLNode RootNode, IXMLNode ParentNode, const InvString
NodeName, const InvString URI, const InvString TypeName, const InvString Value, Boolean GenPre
= False);
```

Description

This is CreateScalarNodeXS, a member of class TSOAPDomConv.

51.1.3.4.23 TSOAPDomConv.CreateTypedNode Method

Pascal

```
function CreateTypedNode(RootNode: IXMLNode; ParentNode: IXMLNode; const NodeName: InvString;
const URI: InvString; TypeName: InvString; GenPre: Boolean = False): IXMLNode;
```

C++

```
__fastcall IXMLNode CreateTypedNode(IXMLNode RootNode, IXMLNode ParentNode, const InvString
NodeName, const InvString URI, InvString TypeName, Boolean GenPre = False);
```

Description

This is CreateTypedNode, a member of class TSOAPDomConv.

51.1.3.4.24 TSOAPDomConv.FinalizeMultiRefNodes Method

Pascal

```
procedure FinalizeMultiRefNodes;
```

C++

```
__fastcall FinalizeMultiRefNodes();
```

Description

This is FinalizeMultiRefNodes, a member of class TSOAPDomConv.

51.1.3.4.25 TSOAPDomConv.FindAttachment Method

Returns a TSOAPAttachment instance that represent the attachment with a specified ID.

Pascal

```
function FindAttachment(const AContentId: string): TSOAPAttachment;
```

C++

```
__fastcall TSOAPAttachment FindAttachment(const UnicodeString AContentId);
```

Description

FindAttachment locates the attachment that is identified by a specified content ID (HREF).

AContentId is the unique identifier for the attachment that was supplied to the AddAttachment method when the attachment was

first added to the multipart form.

FindAttachment returns a TSOAPAttachment instance that represents the attachment.

See Also

AddAttachment (see page 921)

51.1.3.4.26 TSOAPDomConv.FindMultiRefNodeByID Method

Pascal

```
function FindMultiRefNodeByID(const ID: string): Pointer;
```

C++

```
__fastcall void * FindMultiRefNodeByID(const UnicodeString ID);
```

Description

This is FindMultiRefNodeByID, a member of class TSOAPDomConv.

51.1.3.4.27 TSOAPDomConv.FindMultiRefNodeByInstance Method

Pascal

```
function FindMultiRefNodeByInstance(Instance: Pointer): string;
```

C++

```
__fastcall UnicodeString FindMultiRefNodeByInstance(void * Instance);
```

Description

This is FindMultiRefNodeByInstance, a member of class TSOAPDomConv.

51.1.3.4.28 TSOAPDomConv.FindNodeByHREF Method

Pascal

```
function FindNodeByHREF(RootNode: IXMLNode; const HREF: InvString): IXMLNode;
```

C++

```
__fastcall IXMLNode FindNodeByHREF(IXMLNode RootNode, const InvString HREF);
```

Description

This is FindNodeByHREF, a member of class TSOAPDomConv.

51.1.3.4.29 TSOAPDomConv.FindPrefixForURI Method

Pascal

```
function FindPrefixForURI(RootNode: IXMLNode; Node: IXMLNode; const URI: InvString;
DeclIfNone: Boolean = False): InvString;
```

C++

```
__fastcall InvString FindPrefixForURI(IXMLNode RootNode, IXMLNode Node, const InvString URI,
Boolean DeclIfNone = False);
```

Description

This is FindPrefixForURI, a member of class TSOAPDomConv.

51.1.3.4.30 TSOAPDomConv.GetDataNode Method

Pascal

```
function GetDataNode(RootNode: IXMLNode; Node: IXMLNode; var ID: InvString): IXMLNode;
```

C++

```
__fastcall IXMLNode GetDataNode(IXMLNode RootNode, IXMLNode Node, InvString ID);
```

Description

This is GetDataNode, a member of class TSOAPDomConv.

51.1.3.4.31 TSOAPDomConv.GetElementType Method

Pascal

```
function GetElementType(Node: IXMLNode; var TypeURI: InvString; var TypeName: InvString):
Boolean;
```

C++

```
__fastcall Boolean GetElementType(IXMLNode Node, InvString TypeURI, InvString TypeName);
```

Description

This is GetElementType, a member of class TSOAPDomConv.

51.1.3.4.32 TSOAPDomConv.GetNewID Method

Pascal

```
function GetNewID: string;
```

C++

```
__fastcall UnicodeString GetNewID();
```

Description

This is GetNewID, a member of class TSOAPDomConv.

51.1.3.4.33 TSOAPDomConv.GetNodeAsText Method

Pascal

```
function GetNodeAsText(Node: IXMLNode): InvString;
```

C++

```
__fastcall InvString GetNodeAsText(IXMLNode Node);
```

Description

This is GetNodeAsText, a member of class TSOAPDomConv.

51.1.3.4.34 TSOAPDomConv.GetObjectPropAsText Method

Pascal

```
function GetObjectPropAsText(Instance: TObject; PropInfo: PPropInfo): InvString;
```

C++

```
__fastcall InvString GetObjectPropAsText(TObject * Instance, PPropInfo PropInfo);
```

Description

This is GetObjectPropAsText, a member of class TSOAPDomConv.

51.1.3.4.35 TSOAPDomConv.GetOptions Method

Pascal

```
function GetOptions: TSOAPConvertOptions;
```

C++

```
__fastcall TSOAPConvertOptions GetOptions();
```

Description

This is GetOptions, a member of class TSOAPDomConv.

51.1.3.4.36 TSOAPDomConv.GetTypeBySchemaNS Method

Pascal

```
function GetTypeBySchemaNS(Node: IXMLNode; const URI: InvString): Variant;
```

C++

```
__fastcall Variant GetTypeBySchemaNS(IXMLNode Node, const InvString URI);
```

Description

This is GetTypeBySchemaNS, a member of class TSOAPDomConv.

51.1.3.4.37 TSOAPDomConv.InitObjectFromSOAP Method

Initializes an instance of a remotable object to reflect the values in its SOAP representation.

Pascal

```
procedure InitObjectFromSOAP(Instance: TObject; RootNode: IXMLNode; Node: IXMLNode);
```

C++

```
__fastcall InitObjectFromSOAP(TObject * Instance, IXMLNode RootNode, IXMLNode Node);
```

Description

InitObjectFromSOAP assigns values to the published properties of a remotable object that represents a node from the SOAP envelope.

Instance is the remotable object to initialize from its SOAP representation.

RootNode is an ancestor node of the node for the remotable object that can contain information needed for initializing the remotable object.

Node is the node in the SOAP envelope that represents the remotable object.

See Also

ObjInstanceToSOAP (see page 922)

51.1.3.4.38 TSOAPDomConv.IsNodeAVarArray Method

Pascal

```
function IsNodeAVarArray(const Node: IXMLNode; var VT: TVarType): Boolean;
```

C++

```
__fastcall Boolean IsNodeAVarArray(const IXMLNode Node, TVarType VT);
```

Description

This is IsNodeAVarArray, a member of class TSOAPDomConv.

51.1.3.4.39 TSOAPDomConv.IsObjectWriting Method

Pascal

```
function IsObjectWriting(Instance: TObject): Boolean;
```

C++

```
__fastcall Boolean IsObjectWriting(TObject * Instance);
```

Description

This is IsObjectWriting, a member of class TSOAPDomConv.

51.1.3.4.40 TSOAPDomConv.LoadObject Method

Pascal

```
procedure LoadObject(Instance: TObject; RootNode: IXMLNode; Node: IXMLNode);
```

C++

```
__fastcall LoadObject(TObject * Instance, IXMLNode RootNode, IXMLNode Node);
```

Description

This is LoadObject, a member of class TSOAPDomConv.

51.1.3.4.41 MakeArrayNode Method

This is the overview for the MakeArrayNode method overload.

Overload List

	Name	Description
	TSOAPDomConv.MakeArrayNode (IXMLNode, IXMLNode, InvString, InvString, InvString, array of Integer) (see page 2774)	This is MakeArrayNode, a member of class TSOAPDomConv.
	TSOAPDomConv.MakeArrayNode (IXMLNode, IXMLNode, InvString, InvString, InvString, Integer, Integer) (see page 2774)	This is MakeArrayNode, a member of class TSOAPDomConv.

51.1.3.4.41.1 TSOAPDomConv.MakeArrayNode Method (IXMLNode, IXMLNode, InvString, InvString, InvString, Integer, Integer)

Pascal

```
function MakeArrayNode(RootNode: IXMLNode; Node: IXMLNode; const Name: InvString; const URI:
InvString; const TypeName: InvString; Dim: Integer; Len: Integer): IXMLNode; overload;
```

C++

```
__fastcall IXMLNode MakeArrayNode(IXMLNode RootNode, IXMLNode Node, const InvString Name,
const InvString URI, const InvString TypeName, int Dim, int Len);
```

Description

This is MakeArrayNode, a member of class TSOAPDomConv.

51.1.3.4.41.2 TSOAPDomConv.MakeArrayNode Method (IXMLNode, IXMLNode, InvString, InvString, InvString, array of Integer)

Pascal

```
function MakeArrayNode(RootNode: IXMLNode; Node: IXMLNode; const Name: InvString; const URI:
InvString; const TypeName: InvString; Indices: array of Integer): IXMLNode; overload;
```

C++

```
__fastcall IXMLNode MakeArrayNode(IXMLNode RootNode, IXMLNode Node, const InvString Name,
const InvString URI, const InvString TypeName, array of Integer Indices);
```

Description

This is MakeArrayNode, a member of class TSOAPDomConv.

51.1.3.4.42 TSOAPDomConv.MultiRefObject Method

Pascal

```
function MultiRefObject(Cls: TClass): Boolean;
```

C++

```
__fastcall Boolean MultiRefObject(TClass Cls);
```

Description

This is MultiRefObject, a member of class TSOAPDomConv.

51.1.3.4.43 TSOAPDomConv.NodeIsNULL Method

Pascal

```
function NodeIsNULL(Node: IXMLNode): Boolean;
```

C++

```
__fastcall Boolean NodeIsNULL(IXMLNode Node);
```

Description

This is NodeIsNULL, a member of class TSOAPDomConv.

51.1.3.4.44 TSOAPDomConv.ObjectMemberNoShow Method

Pascal

```
procedure ObjectMemberNoShow(const ClassName: string; const MemberName: string);
```

C++

```
__fastcall ObjectMemberNoShow(const UnicodeString ClassName, const UnicodeString MemberName);
```

Description

This is ObjectMemberNoShow, a member of class TSOAPDomConv.

51.1.3.4.45 TSOAPDomConv.ObjInstanceToSOAP Method

Returns the SOAP representation of a remotable object.

Pascal

```
function ObjInstanceToSOAP(Instance: TObject; RootNode: IXMLNode; ParentNode: IXMLNode; const
NodeName: InvString; const NodeNamespace: InvString; ObjConvOpts: TObjectConvertOptions; out
RefID: InvString): IXMLNode;
```

C++

```
__fastcall IXMLNode ObjInstanceToSOAP(TObject * Instance, IXMLNode RootNode, IXMLNode
ParentNode, const InvString NodeName, const InvString NodeNamespace, TObjectConvertOptions
ObjConvOpts, InvString RefID);
```

Description

ObjInstanceToSOAP adds a node to represent the value of a remotable object.

Instance is the remotable object to encode as its SOAP representation.

RootNode is an ancestor of the node to which information about the node can be added as attributes.

ParentNode is the node in the SOAP representation that should act as a parent to the node that ObjInstanceToSOAP generates.

NodeName is the name of the node that is generated.

NodeNamespace is the URI for the namespace in which NodeName is defined.

ObjConvOpts are flags that customize the way the node is serialized.

RefID returns a reference ID for the node. This is used if the node is multi-referenced.

ObjInstanceToSOAP returns the XML for the newly generated node.

See Also

InitObjectFromSOAP (see page 922)

51.1.3.4.46 TSOAPDomConv.ReadHeader Method

Pascal

```
procedure ReadHeader(const EnvNode: IXMLNode; const HdrNode: IXMLNode; Headers: THeaderList);
```

C++

```
__fastcall ReadHeader(const IXMLNode EnvNode, const IXMLNode HdrNode, THeaderList Headers);
```

Description

This is ReadHeader, a member of class TSOAPDomConv.

51.1.3.4.47 TSOAPDomConv.ReadRectDynArray Method

Pascal

```
procedure ReadRectDynArray(RootNode: IXMLNode; Node: IXMLNode; Info: PTypeInfo; Dims: Integer;
P: Pointer; CurElem: Integer);
```

C++

```
__fastcall ReadRectDynArray(IXMLNode RootNode, IXMLNode Node, PTypeInfo Info, int Dims, void *
P, int CurElem);
```

Description

This is ReadRectDynArray, a member of class TSOAPDomConv.

51.1.3.4.48 TSOAPDomConv.ReadRectDynArrayElem Method

Pascal

```
procedure ReadRectDynArrayElem(RootNode: IXMLNode; Node: IXMLNode; Info: PTypeInfo; Size:
Integer; Dim: Integer; P: Pointer; var CurElem: Integer);
```

C++

```
__fastcall ReadRectDynArrayElem(IXMLNode RootNode, IXMLNode Node, PTypeInfo Info, int Size,
int Dim, void * P, int CurElem);
```

Description

This is ReadRectDynArrayElem, a member of class TSOAPDomConv.

51.1.3.4.49 TSOAPDomConv.ReadRow Method

Pascal

```
procedure ReadRow(RootNode: IXMLNode; Node: IXMLNode; var CurElem: Integer; Size: Integer; P:
Pointer; Info: PTypeInfo);
```

C++

```
__fastcall ReadRow(IXMLNode RootNode, IXMLNode Node, int CurElem, int Size, void * P,
PTypeInfo Info);
```

Description

This is ReadRow, a member of class TSOAPDomConv.

51.1.3.4.50 TSOAPDomConv.ReadVarArrayDim Method

Pascal

```
function ReadVarArrayDim(Node: IXMLNode; IsVarVArray: Boolean = False; VT: TVarType = 0):
Variant;
```

C++

```
__fastcall Variant ReadVarArrayDim(IXMLNode Node, Boolean IsVarVArray = False, TVarType VT =
0);
```

Description

This is ReadVarArrayDim, a member of class TSOAPDomConv.

51.1.3.4.51 TSOAPDomConv.ReadVariant Method

Pascal

```
procedure ReadVariant(Node: IXMLNode; P: Pointer);
```

C++

```
__fastcall ReadVariant(IXMLNode Node, void * P);
```

Description

This is ReadVariant, a member of class TSOAPDomConv.

51.1.3.4.52 TSOAPDomConv.RemoveObjectAsWriting Method

Pascal

```
procedure RemoveObjectAsWriting(Instance: TObject);
```

C++

```
__fastcall RemoveObjectAsWriting(TObject * Instance);
```

Description

This is RemoveObjectAsWriting, a member of class TSOAPDomConv.

51.1.3.4.53 TSOAPDomConv.ResetMultiRef Method

Pascal

```
procedure ResetMultiRef;
```

C++

```
__fastcall ResetMultiRef();
```

Description

This is ResetMultiRef, a member of class TSOAPDomConv.

51.1.3.4.54 SerializationOptions Method

This is the overview for the SerializationOptions method overload.

Overload List

	Name	Description
	TSOAPDomConv.SerializationOptions (PTypeInfo) (see page 2778)	This is SerializationOptions, a member of class TSOAPDomConv.
	TSOAPDomConv.SerializationOptions (TClass) (see page 2778)	This is SerializationOptions, a member of class TSOAPDomConv.
	TSOAPDomConv.SerializationOptions (TObject) (see page 2778)	This is SerializationOptions, a member of class TSOAPDomConv.

51.1.3.4.54.1 TSOAPDomConv.SerializationOptions Method (PTypeInfo)

Pascal

```
function SerializationOptions(ATypeInfo: PTypeInfo): TSerializationOptions; overload;
```

C++

```
__fastcall TSerializationOptions SerializationOptions(PTypeInfo ATypeInfo);
```

Description

This is SerializationOptions, a member of class TSOAPDomConv.

51.1.3.4.54.2 TSOAPDomConv.SerializationOptions Method (TClass)

Pascal

```
function SerializationOptions(Cls: TClass): TSerializationOptions; overload;
```

C++

```
__fastcall TSerializationOptions SerializationOptions(TClass Cls);
```

Description

This is SerializationOptions, a member of class TSOAPDomConv.

51.1.3.4.54.3 TSOAPDomConv.SerializationOptions Method (TObject)

Pascal

```
function SerializationOptions(Obj: TObject): TSerializationOptions; overload;
```

C++

```
__fastcall TSerializationOptions SerializationOptions(TObject * Obj);
```

Description

This is SerializationOptions, a member of class TSOAPDomConv.

51.1.3.4.55 TSOAPDomConv.SetNodeType Method

Pascal

```
procedure SetNodeType(RootNode: IXMLNode; InstNode: IXMLNode; const ElemURI: InvString; const
TypeName: InvString);
```

C++

```
__fastcall SetNodeType(IXMLNode RootNode, IXMLNode InstNode, const InvString ElemURI, const
InvString TypeName);
```

Description

This is SetNodeType, a member of class TSOAPDomConv.

51.1.3.4.56 TSOAPDomConv.SetObjectPropFromText Method

Pascal

```
procedure SetObjectPropFromText(Instance: TObject; PropInfo: PPropInfo; const SoapData:
InvString);
```

C++

```
__fastcall SetObjectPropFromText(TObject * Instance, PPropInfo PropInfo, const InvString
SoapData);
```

Description

This is SetObjectPropFromText, a member of class TSOAPDomConv.

51.1.3.4.57 TSOAPDomConv.SetOptions Method

Pascal

```
procedure SetOptions(const Value: TSOAPConvertOptions);
```

C++

```
__fastcall SetOptions(const TSOAPConvertOptions Value);
```

Description

This is SetOptions, a member of class TSOAPDomConv.

51.1.3.4.58 TSOAPDomConv.UnhandledNode Method

Pascal

```
procedure UnhandledNode(const Name: string; NodeXML: InvString);
```

C++

```
__fastcall UnhandledNode(const UnicodeString Name, InvString NodeXML);
```

Description

This is UnhandledNode, a member of class TSOAPDomConv.

51.1.3.4.59 TSOAPDomConv.WriteHeader Method

Pascal

```
procedure WriteHeader(const Header: TObject; RootNode: IXMLNode; ParentNode: IXMLNode);
```

C++

```
__fastcall WriteHeader(const TObject * Header, IXMLNode RootNode, IXMLNode ParentNode);
```

Description

This is WriteHeader, a member of class TSOAPDomConv.

51.1.3.4.60 TSOAPDomConv.WriteNonRectDynArray Method

Pascal

```
procedure WriteNonRectDynArray(RootNode: IXMLNode; Node: IXMLNode; const Name: InvString;
Info: PTypeInfo; const URI: InvString; const TypeName: InvString; P: Pointer; Dim: Integer);
```

C++

```
__fastcall WriteNonRectDynArray(IXMLNode RootNode, IXMLNode Node, const InvString Name,
PTypeInfo Info, const InvString URI, const InvString TypeName, void * P, int Dim);
```

Description

This is WriteNonRectDynArray, a member of class TSOAPDomConv.

51.1.3.4.61 TSOAPDomConv.WriteNonRectDynArrayElem Method

Pascal

```
function WriteNonRectDynArrayElem(RootNode: IXMLNode; Node: IXMLNode; Info: PTypeInfo; const
URI: InvString; const TypeName: InvString; P: Pointer; Dim: Integer): Integer;
```

C++

```
__fastcall int WriteNonRectDynArrayElem(IXMLNode RootNode, IXMLNode Node, PTypeInfo Info,
const InvString URI, const InvString TypeName, void * P, int Dim);
```

Description

This is WriteNonRectDynArrayElem, a member of class TSOAPDomConv.

51.1.3.4.62 TSOAPDomConv.WriteRectDynArray Method

Pascal

```
procedure WriteRectDynArray(RootNode: IXMLNode; Node: IXMLNode; Info: PTypeInfo; Dims:
Integer; P: Pointer; const TypeName: InvString);
```

C++

```
__fastcall WriteRectDynArray(IXMLNode RootNode, IXMLNode Node, PTypeInfo Info, int Dims, void
* P, const InvString TypeName);
```

Description

This is WriteRectDynArray, a member of class TSOAPDomConv.

51.1.3.4.63 TSOAPDomConv.WriteRectDynArrayElem Method

Pascal

```
procedure WriteRectDynArrayElem(RootNode: IXMLNode; Node: IXMLNode; Info: PTypeInfo; Size:
Integer; Dim: Integer; P: Pointer; const TypeName: InvString);
```

C++

```
__fastcall WriteRectDynArrayElem(IXMLNode RootNode, IXMLNode Node, PTypeInfo Info, int Size,
int Dim, void * P, const InvString TypeName);
```

Description

This is WriteRectDynArrayElem, a member of class TSOAPDomConv.

51.1.3.4.64 TSOAPDomConv.WriteVarArray Method

Pascal

```
procedure WriteVarArray(RootNode: IXMLNode; Node: IXMLNode; const Name: InvString; V: Variant);
```

C++

```
__fastcall WriteVarArray(IXMLNode RootNode, IXMLNode Node, const InvString Name, Variant V);
```

Description

This is WriteVarArray, a member of class TSOAPDomConv.

51.1.3.4.65 TSOAPDomConv.WriteVarArrayAsB64 Method

Pascal

```
procedure WriteVarArrayAsB64(RootNode: IXMLNode; Node: IXMLNode; const Name: InvString; V:
Variant);
```

C++

```
__fastcall WriteVarArrayAsB64(IXMLNode RootNode, IXMLNode Node, const InvString Name, Variant
V);
```

Description

This is WriteVarArrayAsB64, a member of class TSOAPDomConv.

51.1.3.4.66 TSOAPDomConv.WriteVariant Method

Pascal

```
procedure WriteVariant(RootNode: IXMLNode; Node: IXMLNode; const Name: InvString; V: Variant);
```

C++

```
__fastcall WriteVariant(IXMLNode RootNode, IXMLNode Node, const InvString Name, Variant V);
```

Description

This is WriteVariant, a member of class TSOAPDomConv.

51.1.3.5 TSOAPDomConv Properties

The properties of the TSOAPDomConv class are listed here.

Published Properties

	Name	Description
	Options (see page 2781)	Controls how the TSOAPDomConv (see page 2758) object performs certain types of conversions.

51.1.3.5.1 TSOAPDomConv.Options Property

Controls how the TSOAPDomConv (see page 2758) object performs certain types of conversions.

Pascal

```
property Options: TSOAPConvertOptions;
```

C++

```
__property TSOAPConvertOptions Options;
```

Description

Set Options to indicate how you want the TSOAPDomConv (see page 2758) object to handle various types of conversions. This set can include zero or more of the TSOAPConvertOption (see page 2740) flags.

See Also

ConvertNativeDataToSoap (see page 2765)

ClassOptions (see page 877)

ConvertSoapToNativeData (see page 2766)

51.1.3.6 TSOAPDomConv Events

The events of the TSOAPDomConv class are listed here.

Published Events

	Name	Description
	OnMemberDataNotReceived (see page 2782)	Occurs when converting a SOAP representation to a remotable object and an expected class member was not found.
	OnUnhandledNode (see page 2782)	Occurs when a SOAP representation includes an XML node that the converter does not know how to handle.

51.1.3.6.1 TSOAPDomConv.OnMemberDataNotReceived Event

Occurs when converting a SOAP representation to a remotable object and an expected class member was not found.

Pascal

```
property OnMemberDataNotReceived: TMemberDataNotReceivedEvent;
```

C++

```
__property TMemberDataNotReceivedEvent OnMemberDataNotReceived;
```

Description

Write an OnMemberDataNotReceived to detect cases where the converter does not receive the information it excepts When a method on an invokable interface requires the user to send or receive OnMemberDataNotReceived, these for deserializing a remotable type (a TRemotable (see page 868) descendant). This event is especially useful during debugging. Typically, after detecting a problem using OnMemberDataNotReceived, the problem is then handled by changing the code that declares or registers the remotable type.

See Also

TRemotable (see page 868)

51.1.3.6.2 TSOAPDomConv.OnUnhandledNode Event

Occurs when a SOAP representation includes an XML node that the converter does not know how to handle.

Pascal

```
property OnUnhandledNode: TUnhandledNodeEvent;
```

C++

```
__property TUnhandledNodeEvent OnUnhandledNode;
```

Description

Write an OnUnhandledNode to detect and, possibly, fix cases where the converter can't correctly generate the native code corresponding to a node in a SOAP packet.

51.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	GetOrdPropEx (see page 2783)	This is function OPToSOAPDomConv.GetOrdPropEx.

51.2.1 OPToSOAPDomConv.GetOrdPropEx Function

Pascal

```
function GetOrdPropEx(Instance: TObject; PropInfo: PPropInfo): Longint;
```

C++

```
Longint GetOrdPropEx(TObject * Instance, PPropInfo PropInfo);
```

File

OPToSOAPDomConv

Description

This is function OPToSOAPDomConv.GetOrdPropEx.

51.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	ConvNodeState (see page 2784)	This is record OPToSOAPDomConv.ConvNodeState.

Records

	Name	Description
	TMultiRefNodeElem (see page 2784)	This is record OPToSOAPDomConv.TMultiRefNodeElem.
	TMultiRefNodeMapElem (see page 2785)	This is record OPToSOAPDomConv.TMultiRefNodeMapElem.
	TSOAPArrayElemDesc (see page 2785)	This is record OPToSOAPDomConv.TSOAPArrayElemDesc.

51.3.1 OPToSOAPDomConv.ConvNodeState Enumeration

Pascal

```
ConvNodeState = (
  nsClientSend,
  nsServerReceive,
  nsServerSend,
  nsClientReceive
);
```

C++

```
enum ConvNodeState {
  nsClientSend,
  nsServerReceive,
  nsServerSend,
  nsClientReceive
};
```

File

OPToSOAPDomConv

Description

This is record OPToSOAPDomConv.ConvNodeState.

51.3.2 OPToSOAPDomConv.TMultiRefNodeElem Record

Pascal

```
TMultiRefNodeElem = record
  Node: IXMLNode;
  MultiRefChildren: TXMLNodeArray;
end;
```

C++

```
struct TMultiRefNodeElem {
  IXMLNode Node;
  TXMLNodeArray MultiRefChildren;
};
```

File

OPToSOAPDomConv

Description

This is record OPToSOAPDomConv.TMultiRefNodeElem.

51.3.3 OPToSOAPDomConv.TMultiRefNodeMapElem Record

Pascal

```
TMultiRefNodeMapElem = record
  Instance: Pointer;
  ID: string;
end;
```

C++

```
struct TMultiRefNodeMapElem {
  void * Instance;
  UnicodeString ID;
};
```

File

OPToSOAPDomConv

Description

This is record OPToSOAPDomConv.TMultiRefNodeMapElem.

51.3.4 OPToSOAPDomConv.TSOAPArrayElemDesc Record

Pascal

```
TSOAPArrayElemDesc = record
  MultiDim: Boolean;
  Dims: TIntegerDynArray;
end;
```

C++

```
struct TSOAPArrayElemDesc {
  Boolean MultiDim;
  TIntegerDynArray Dims;
};
```

File

OPToSOAPDomConv

Description

This is record OPToSOAPDomConv.TSOAPArrayElemDesc.

51.4 Types

The following table lists types in this documentation.

Types

Name	Description
TMemberDataNotReceivedEvent (see page 2786)	TMemberDataNotReceivedEvent is the type of the OnMemberDataNotReceived event of TSOAPDomConv (see page 2758).
TMultiRefNodeMap (see page 2787)	This is type OPToSOAPDomConv.TMultiRefNodeMap.
TMultiRefNodes (see page 2787)	This is type OPToSOAPDomConv.TMultiRefNodes.
TSOAPArrayDesc (see page 2787)	This is type OPToSOAPDomConv.TSOAPArrayDesc.
TUnhandledNodeEvent (see page 2788)	TUnhandledNodeEvent is the type of the OnUnhandledNode event of TSOAPDomConv (see page 2758).
TXMLNodeArray (see page 2788)	This is type OPToSOAPDomConv.TXMLNodeArray.

51.4.1 OPToSOAPDomConv.TMemberDataNotReceivedEvent Type

TMemberDataNotReceivedEvent is the type of the OnMemberDataNotReceived event of TSOAPDomConv (see page 2758).

Pascal

```
TMemberDataNotReceivedEvent = procedure (const ClassName: string; const Member: string) of object;
```

C++

```
(const ClassName: string; const Member: string) ( TMemberDataNotReceivedEvent)();
```

File

OPToSOAPDomConv

Description

TMemberDataNotReceivedEvent is the type of the OnMemberDataNotReceived event handler, which is called when a converter is constructing a remotable object from its SOAP representation and fails to find a published class member.

ClassName is the name of the remotable object's class.

Member is the name of the member that was not found in the SOAP representation.

See Also

Procedural types

Creating events

51.4.2 OPToSOAPDomConv.TMultiRefNodeMap Type

Pascal

```
TMultiRefNodeMap = array of TMultiRefNodeMapElem;
```

C++

```
array of TMultiRefNodeMapElem TMultiRefNodeMap;
```

File

OPToSOAPDomConv

Description

This is type OPToSOAPDomConv.TMultiRefNodeMap.

51.4.3 OPToSOAPDomConv.TMultiRefNodes Type

Pascal

```
TMultiRefNodes = array of TMultiRefNodeElem;
```

C++

```
array of TMultiRefNodeElem TMultiRefNodes;
```

File

OPToSOAPDomConv

Description

This is type OPToSOAPDomConv.TMultiRefNodes.

51.4.4 OPToSOAPDomConv.TSOAPArrayDesc Type

Pascal

```
TSOAPArrayDesc = array of TSOAPArrayElemDesc;
```

C++

```
array of TSOAPArrayElemDesc TSOAPArrayDesc;
```

File

OPToSOAPDomConv

Description

This is type OPToSOAPDomConv.TSOAPArrayDesc.

51.4.5 OPToSOAPDomConv.TUnhandledNodeEvent Type

TUnhandledNodeEvent is the type of the OnUnhandledNode event of TSOAPDomConv (see page 2758).

Pascal

```
TUnhandledNodeEvent = procedure (const Name: string; NodeXML: InvString) of object;
```

C++

```
(const Name: string; NodeXML: InvString) ( TUnhandledNodeEvent)();
```

File

OPToSOAPDomConv

Description

TUnhandledNodeEvent is the type of the OnUnhandledNode event handler, which is called when a converter encounters an XML node that it could not convert.

Name is the name of the XML node that the converter did not recognize.

Node is the interface of the node that the converter did not recognize.

See Also

Procedural types

Creating events

51.4.6 OPToSOAPDomConv.TXMLNodeArray Type

Pascal

```
TXMLNodeArray = array of IXMLNode;
```

C++

```
array of IXMLNode TXMLNodeArray;
```

File

OPToSOAPDomConv

Description

This is type OPToSOAPDomConv.TXMLNodeArray.

51.5 Variables

The following table lists variables in this documentation.

Variables

Name	Description
DefArrayElemName (see page 2789)	This is variable OPToSOAPDomConv.DefArrayElemName.

51.5.1 OPToSOAPDomConv.DefArrayElemName Variable

Pascal

```
DefArrayElemName: string = 'item';
```

C++

```
UnicodeString DefArrayElemName = 'item';
```

File

OPToSOAPDomConv

Description

This is variable OPToSOAPDomConv.DefArrayElemName.

51.6 Constants

The following table lists constants in this documentation.

Constants

Name	Description
SVarArrayType (see page 2790)	This is constant OPToSOAPDomConv.SVarArrayType.

51.6.1 OPToSOAPDomConv.SVarArrayType Constant

Pascal

```
SVarArrayType = 'VarArrayType';
```

C++

```
SVarArrayType = 'VarArrayType';
```

File

OPToSOAPDomConv

Description

This is constant OPToSOAPDomConv.SVarArrayType.

52 OrbPas Namespace

This is namespace OrbPas.

Topics

Name	Description
Interfaces (see page 2793)	The following table lists interfaces in this documentation.
Functions (see page 2821)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 2826)	The following table lists structs, records, enums in this documentation.
Types (see page 2828)	The following table lists types in this documentation.

Functions

	Name	Description
	AnyToVariant (see page 2821)	This is function OrbPas.AnyToVariant.
	BindStub (see page 2821)	This is function OrbPas.BindStub.
	CorbaAnyType (see page 2822)	This is function OrbPas.CorbaAnyType.
	CorbaDuplicateAny (see page 2822)	This is function OrbPas.CorbaDuplicateAny.
	CorbaReleaseAny (see page 2822)	This is function OrbPas.CorbaReleaseAny.
	CorbaStringFree (see page 2823)	This is function OrbPas.CorbaStringFree.
	CorbaWStringFree (see page 2823)	This is function OrbPas.CorbaWStringFree.
	CreateSkeleton (see page 2823)	This is function OrbPas.CreateSkeleton.
	CreateUserException (see page 2824)	This is function OrbPas.CreateUserException.
	InitORB (see page 2824)	This is function OrbPas.InitORB.
	RegisterUserException (see page 2824)	This is function OrbPas.RegisterUserException.
	SequenceToVariant (see page 2825)	This is function OrbPas.SequenceToVariant.
	UnRegisterUserException (see page 2825)	This is function OrbPas.UnRegisterUserException.
	VariantToAny (see page 2825)	This is function OrbPas.VariantToAny.

Interfaces

	Name	Description
	IBOA (see page 2793)	This is class OrbPas.IBOA.
	ICorbaObj (see page 2795)	This is class OrbPas.ICorbaObj.
	IMarshalInBuffer (see page 2796)	This is class OrbPas.IMarshalInBuffer.
	IMarshalOutBuffer (see page 2800)	This is class OrbPas.IMarshalOutBuffer.
	IORB (see page 2805)	This is class OrbPas.IORB.
	ISkeleton (see page 2809)	This is class OrbPas.ISkeleton.
	ISkeletonObject (see page 2810)	This is class OrbPas.ISkeletonObject.

	IStub (see page 2812)	This is class OrbPas.IStub.
	IStubObject (see page 2815)	This is class OrbPas.IStubObject.
	ITypeCode (see page 2816)	This is class OrbPas.ITypeCode.

Structs, Records, Enums

	Name	Description
	TCKind (see page 2826)	This is record OrbPas.TCKind.
	TStructMember (see page 2827)	This is record OrbPas.TStructMember.

Types

Name	Description
CorbaBoolean (see page 2828)	This is type OrbPas.CorbaBoolean.
CorbaULong (see page 2828)	This is type OrbPas.CorbaULong.
PCorbaAny (see page 2829)	This is type OrbPas.PCorbaAny.
PCorbaInterfaceDef (see page 2829)	This is type OrbPas.PCorbaInterfaceDef.
PCorbaObject (see page 2829)	This is type OrbPas.PCorbaObject.
PCorbaOrb (see page 2830)	This is type OrbPas.PCorbaOrb.
PExceptionDescription (see page 2830)	This is type OrbPas.PExceptionDescription.
PMarshalInbuffer (see page 2830)	This is type OrbPas.PMarshalInbuffer.
PMarshalOutbuffer (see page 2831)	This is type OrbPas.PMarshalOutbuffer.
PUserExceptionProxy (see page 2831)	This is type OrbPas.PUserExceptionProxy.
TArgv (see page 2831)	This is type OrbPas.TArgv.
TCopyUserExceptionProc (see page 2832)	This is type OrbPas.TCopyUserExceptionProc.
TStructMembers (see page 2832)	This is type OrbPas.TStructMembers.
TThrowUserExceptionProc (see page 2832)	This is type OrbPas.TThrowUserExceptionProc.
TUserExceptionFactoryProc (see page 2833)	This is type OrbPas.TUserExceptionFactoryProc.

52.1 Interfaces

The following table lists interfaces in this documentation.

Interfaces

	Name	Description
	IBOA (see page 2793)	This is class OrbPas.IBOA.
	ICorbaObj (see page 2795)	This is class OrbPas.ICorbaObj.
	IMarshalInBuffer (see page 2796)	This is class OrbPas.IMarshalInBuffer.
	IMarshalOutBuffer (see page 2800)	This is class OrbPas.IMarshalOutBuffer.
	IORB (see page 2805)	This is class OrbPas.IORB.
	ISkeleton (see page 2809)	This is class OrbPas.ISkeleton.
	ISkeletonObject (see page 2810)	This is class OrbPas.ISkeletonObject.
	IStub (see page 2812)	This is class OrbPas.IStub.
	IStubObject (see page 2815)	This is class OrbPas.IStubObject.
	ITypeCode (see page 2816)	This is class OrbPas.ITypeCode.

52.1.1 IBOA Interface

Class Hierarchy

OrbPas.IBOA

Pascal

```
IBOA = interface;
```

C++

```
__interface IBOA;
```

File

OrbPas

Description

This is class OrbPas.IBOA.

52.1.1.1 IBOA Members

The following tables list the members exposed by IBOA.

Public Methods

	Name	Description
	Deactivate (see page 2794)	This is Deactivate, a member of class IBOA.
	GetPrincipal (see page 2794)	This is GetPrincipal, a member of class IBOA.
	GetPrincipalLength (see page 2794)	This is GetPrincipalLength, a member of class IBOA.
	ImplIsReady (see page 2794)	This is ImplIsReady, a member of class IBOA.
	ObjIsReady (see page 2795)	This is ObjIsReady, a member of class IBOA.

52.1.1.2 IBOA Methods

The methods of the IBOA class are listed here.

Public Methods

	Name	Description
	Deactivate (see page 2794)	This is Deactivate, a member of class IBOA.
	GetPrincipal (see page 2794)	This is GetPrincipal, a member of class IBOA.
	GetPrincipalLength (see page 2794)	This is GetPrincipalLength, a member of class IBOA.
	ImplIsReady (see page 2794)	This is ImplIsReady, a member of class IBOA.
	ObjIsReady (see page 2795)	This is ObjIsReady, a member of class IBOA.

52.1.1.2.1 IBOA.Deactivate Method

Pascal

```
procedure Deactivate(const Skeleton: ISkeleton); stdcall;
```

C++

```
__fastcall __stdcall Deactivate(const ISkeleton Skeleton);
```

Description

This is Deactivate, a member of class IBOA.

52.1.1.2.2 IBOA.GetPrincipal Method

Pascal

```
function GetPrincipal(Skeleton: ISkeleton; Bytes: Pointer): Integer; stdcall;
```

C++

```
__fastcall __stdcall int GetPrincipal(ISkeleton Skeleton, void * Bytes);
```

Description

This is GetPrincipal, a member of class IBOA.

52.1.1.2.3 IBOA.GetPrincipalLength Method

Pascal

```
function GetPrincipalLength(Skeleton: ISkeleton): Integer; stdcall;
```

C++

```
__fastcall __stdcall int GetPrincipalLength(ISkeleton Skeleton);
```

Description

This is GetPrincipalLength, a member of class IBOA.

52.1.1.2.4 IBOA.ImplIsReady Method

Pascal

```
procedure ImplIsReady; stdcall;
```

C++

```
__fastcall __stdcall ImplIsReady();
```

Description

This is ImplIsReady, a member of class IBOA.

52.1.1.2.5 IBOA.ObjIsReady Method

Pascal

```
procedure ObjIsReady(const Skeleton: ISkeleton); stdcall;
```

C++

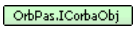
```
__fastcall __stdcall ObjIsReady(const ISkeleton Skeleton);
```

Description

This is ObjIsReady, a member of class IBOA.

52.1.2 ICorbaObj Interface

Class Hierarchy



Pascal

```
ICorbaObj = interface;
```

C++

```
__interface ICorbaObj;
```

File

OrbPas

Description

This is class OrbPas.ICorbaObj.

52.1.2.1 ICorbaObj Members

The following tables list the members exposed by ICorbaObj.

Public Methods

	Name	Description
	CorbaObject (see page 2796)	This is CorbaObject, a member of class ICorbaObj.
	IsLocal (see page 2796)	This is IsLocal, a member of class ICorbaObj.

52.1.2.2 ICorbaObj Methods

The methods of the ICorbaObj class are listed here.

Public Methods

	Name	Description
	CorbaObject (see page 2796)	This is CorbaObject, a member of class ICorbaObj.
	IsLocal (see page 2796)	This is IsLocal, a member of class ICorbaObj.

52.1.2.2.1 ICorbaObj.CorbaObject Method

Pascal

```
function CorbaObject: PCorbaObject; stdcall;
```

C++

```
__fastcall __stdcall PCorbaObject CorbaObject();
```

Description

This is CorbaObject, a member of class ICorbaObj.

52.1.2.2.2 ICorbaObj.IsLocal Method

Pascal

```
function IsLocal: CorbaBoolean; stdcall;
```

C++

```
__fastcall __stdcall CorbaBoolean IsLocal();
```

Description

This is IsLocal, a member of class ICorbaObj.

52.1.3 IMarshalInBuffer Interface

Class Hierarchy

OrbPas.IMarshalInBuffer

Pascal

```
IMarshalInBuffer = interface;
```

C++

```
__interface IMarshalInBuffer;
```

File

OrbPas

Description

This is class OrbPas.IMarshalInBuffer.

52.1.3.1 IMarshalInBuffer Members

The following tables list the members exposed by IMarshalInBuffer.

Public Methods

	Name	Description
	Buffer (see page 2797)	This is Buffer, a member of class IMarshalInBuffer.
	GetAny (see page 2798)	This is GetAny, a member of class IMarshalInBuffer.
	GetChar (see page 2798)	This is GetChar, a member of class IMarshalInBuffer.
	GetDouble (see page 2798)	This is GetDouble, a member of class IMarshalInBuffer.
	GetFloat (see page 2798)	This is GetFloat, a member of class IMarshalInBuffer.
	GetLong (see page 2798)	This is GetLong, a member of class IMarshalInBuffer.
	GetObject (see page 2799)	This is GetObject, a member of class IMarshalInBuffer.
	GetShort (see page 2799)	This is GetShort, a member of class IMarshalInBuffer.
	GetText (see page 2799)	This is GetText, a member of class IMarshalInBuffer.
	GetUnsignedChar (see page 2799)	This is GetUnsignedChar, a member of class IMarshalInBuffer.
	GetUnsignedLong (see page 2800)	This is GetUnsignedLong, a member of class IMarshalInBuffer.
	GetUnsignedShort (see page 2800)	This is GetUnsignedShort, a member of class IMarshalInBuffer.
	GetWidechar (see page 2800)	This is GetWidechar, a member of class IMarshalInBuffer.
	GetWideText (see page 2800)	This is GetWideText, a member of class IMarshalInBuffer.

52.1.3.2 IMarshalInBuffer Methods

The methods of the IMarshalInBuffer class are listed here.

Public Methods

	Name	Description
	Buffer (see page 2797)	This is Buffer, a member of class IMarshalInBuffer.
	GetAny (see page 2798)	This is GetAny, a member of class IMarshalInBuffer.
	GetChar (see page 2798)	This is GetChar, a member of class IMarshalInBuffer.
	GetDouble (see page 2798)	This is GetDouble, a member of class IMarshalInBuffer.
	GetFloat (see page 2798)	This is GetFloat, a member of class IMarshalInBuffer.
	GetLong (see page 2798)	This is GetLong, a member of class IMarshalInBuffer.
	GetObject (see page 2799)	This is GetObject, a member of class IMarshalInBuffer.
	GetShort (see page 2799)	This is GetShort, a member of class IMarshalInBuffer.
	GetText (see page 2799)	This is GetText, a member of class IMarshalInBuffer.
	GetUnsignedChar (see page 2799)	This is GetUnsignedChar, a member of class IMarshalInBuffer.
	GetUnsignedLong (see page 2800)	This is GetUnsignedLong, a member of class IMarshalInBuffer.
	GetUnsignedShort (see page 2800)	This is GetUnsignedShort, a member of class IMarshalInBuffer.
	GetWidechar (see page 2800)	This is GetWidechar, a member of class IMarshalInBuffer.
	GetWideText (see page 2800)	This is GetWideText, a member of class IMarshalInBuffer.

52.1.3.2.1 IMarshalInBuffer.Buffer Method

Pascal

```
function Buffer: PMarshalInbuffer; stdcall;
```

C++

```
__fastcall __stdcall PMarshalInbuffer Buffer();
```

Description

This is Buffer, a member of class IMarshalInBuffer.

52.1.3.2.2 IMarshalInBuffer.GetAny Method

Pascal

```
function GetAny: PCorbaAny; stdcall;
```

C++

```
__fastcall __stdcall PCorbaAny GetAny();
```

Description

This is GetAny, a member of class IMarshalInBuffer.

52.1.3.2.3 IMarshalInBuffer.GetChar Method

Pascal

```
function GetChar: Shortint; stdcall;
```

C++

```
__fastcall __stdcall Shortint GetChar();
```

Description

This is GetChar, a member of class IMarshalInBuffer.

52.1.3.2.4 IMarshalInBuffer.GetDouble Method

Pascal

```
function GetDouble: Double; stdcall;
```

C++

```
__fastcall __stdcall Double GetDouble();
```

Description

This is GetDouble, a member of class IMarshalInBuffer.

52.1.3.2.5 IMarshalInBuffer.GetFloat Method

Pascal

```
function GetFloat: Single; stdcall;
```

C++

```
__fastcall __stdcall Single GetFloat();
```

Description

This is GetFloat, a member of class IMarshalInBuffer.

52.1.3.2.6 IMarshalInBuffer.GetLong Method

Pascal

```
function GetLong: Integer; stdcall;
```

C++

```
__fastcall __stdcall int GetLong();
```

Description

This is GetLong, a member of class IMarshalInBuffer.

52.1.3.2.7 IMarshalInBuffer.GetObject Method

Pascal

```
procedure GetObject(out Obj: ICorbaObj; ref_cnt: CorbaBoolean = True); stdcall;
```

C++

```
__fastcall __stdcall GetObject(ICorbaObj Obj, CorbaBoolean ref_cnt = True);
```

Description

This is GetObject, a member of class IMarshalInBuffer.

52.1.3.2.8 IMarshalInBuffer.GetShort Method

Pascal

```
function GetShort: Smallint; stdcall;
```

C++

```
__fastcall __stdcall Smallint GetShort();
```

Description

This is GetShort, a member of class IMarshalInBuffer.

52.1.3.2.9 IMarshalInBuffer.GetText Method

Pascal

```
function GetText: PChar; stdcall;
```

C++

```
__fastcall __stdcall const char * GetText();
```

Description

This is GetText, a member of class IMarshalInBuffer.

52.1.3.2.10 IMarshalInBuffer.GetUnsignedChar Method

Pascal

```
function GetUnsignedChar: Byte; stdcall;
```

C++

```
__fastcall __stdcall Byte GetUnsignedChar();
```

Description

This is GetUnsignedChar, a member of class IMarshalInBuffer.

52.1.3.2.11 IMarshalInBuffer.GetUnsignedLong Method

Pascal

```
function GetUnsignedLong: UINT; stdcall;
```

C++

```
__fastcall __stdcall UINT GetUnsignedLong();
```

Description

This is GetUnsignedLong, a member of class IMarshalInBuffer.

52.1.3.2.12 IMarshalInBuffer.GetUnsignedShort Method

Pascal

```
function GetUnsignedShort: Word; stdcall;
```

C++

```
__fastcall __stdcall Word GetUnsignedShort();
```

Description

This is GetUnsignedShort, a member of class IMarshalInBuffer.

52.1.3.2.13 IMarshalInBuffer.GetWidechar Method

Pascal

```
function GetWidechar: Word; stdcall;
```

C++

```
__fastcall __stdcall Word GetWidechar();
```

Description

This is GetWidechar, a member of class IMarshalInBuffer.

52.1.3.2.14 IMarshalInBuffer.GetWideText Method

Pascal

```
function GetWideText: PWideChar; stdcall;
```

C++

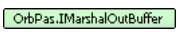
```
__fastcall __stdcall PWideChar GetWideText();
```

Description

This is GetWideText, a member of class IMarshalInBuffer.

52.1.4 IMarshalOutBuffer Interface

Class Hierarchy



Pascal

```
IMarshalOutBuffer = interface;
```

C++

```
__interface IMarshalOutBuffer;
```

File

OrbPas

Description

This is class OrbPas.IMarshalOutBuffer.

52.1.4.1 IMarshalOutBuffer Members

The following tables list the members exposed by IMarshalOutBuffer.

Public Methods

	Name	Description
	Buffer (see page 2802)	This is Buffer, a member of class IMarshalOutBuffer.
	PutAny (see page 2802)	This is PutAny, a member of class IMarshalOutBuffer.
	PutChar (see page 2802)	This is PutChar, a member of class IMarshalOutBuffer.
	PutDouble (see page 2802)	This is PutDouble, a member of class IMarshalOutBuffer.
	PutFloat (see page 2802)	This is PutFloat, a member of class IMarshalOutBuffer.
	PutLong (see page 2803)	This is PutLong, a member of class IMarshalOutBuffer.
	PutObject (see page 2803)	This is PutObject, a member of class IMarshalOutBuffer.
	PutShort (see page 2803)	This is PutShort, a member of class IMarshalOutBuffer.
	PutText (see page 2803)	This is PutText, a member of class IMarshalOutBuffer.
	PutUnsignedChar (see page 2804)	This is PutUnsignedChar, a member of class IMarshalOutBuffer.
	PutUnsignedLong (see page 2804)	This is PutUnsignedLong, a member of class IMarshalOutBuffer.
	PutUnsignedShort (see page 2804)	This is PutUnsignedShort, a member of class IMarshalOutBuffer.
	PutWidechar (see page 2804)	This is PutWidechar, a member of class IMarshalOutBuffer.
	PutWideText (see page 2804)	This is PutWideText, a member of class IMarshalOutBuffer.

52.1.4.2 IMarshalOutBuffer Methods

The methods of the IMarshalOutBuffer class are listed here.

Public Methods

	Name	Description
	Buffer (see page 2802)	This is Buffer, a member of class IMarshalOutBuffer.
	PutAny (see page 2802)	This is PutAny, a member of class IMarshalOutBuffer.
	PutChar (see page 2802)	This is PutChar, a member of class IMarshalOutBuffer.
	PutDouble (see page 2802)	This is PutDouble, a member of class IMarshalOutBuffer.
	PutFloat (see page 2802)	This is PutFloat, a member of class IMarshalOutBuffer.
	PutLong (see page 2803)	This is PutLong, a member of class IMarshalOutBuffer.
	PutObject (see page 2803)	This is PutObject, a member of class IMarshalOutBuffer.
	PutShort (see page 2803)	This is PutShort, a member of class IMarshalOutBuffer.
	PutText (see page 2803)	This is PutText, a member of class IMarshalOutBuffer.
	PutUnsignedChar (see page 2804)	This is PutUnsignedChar, a member of class IMarshalOutBuffer.
	PutUnsignedLong (see page 2804)	This is PutUnsignedLong, a member of class IMarshalOutBuffer.
	PutUnsignedShort (see page 2804)	This is PutUnsignedShort, a member of class IMarshalOutBuffer.
	PutWidechar (see page 2804)	This is PutWidechar, a member of class IMarshalOutBuffer.

	PutWideText (see page 2804)	This is PutWideText, a member of class IMarshalOutBuffer.

52.1.4.2.1 IMarshalOutBuffer.Buffer Method

Pascal

```
function Buffer: PMarshalOutbuffer; stdcall;
```

C++

```
__fastcall __stdcall PMarshalOutbuffer Buffer();
```

Description

This is Buffer, a member of class IMarshalOutBuffer.

52.1.4.2.2 IMarshalOutBuffer.PutAny Method

Pascal

```
procedure PutAny(Value: PCorbaAny); stdcall;
```

C++

```
__fastcall __stdcall PutAny(PCorbaAny Value);
```

Description

This is PutAny, a member of class IMarshalOutBuffer.

52.1.4.2.3 IMarshalOutBuffer.PutChar Method

Pascal

```
procedure PutChar(Value: Shortint); stdcall;
```

C++

```
__fastcall __stdcall PutChar(Shortint Value);
```

Description

This is PutChar, a member of class IMarshalOutBuffer.

52.1.4.2.4 IMarshalOutBuffer.PutDouble Method

Pascal

```
procedure PutDouble(Value: Double); stdcall;
```

C++

```
__fastcall __stdcall PutDouble(Double Value);
```

Description

This is PutDouble, a member of class IMarshalOutBuffer.

52.1.4.2.5 IMarshalOutBuffer.PutFloat Method

Pascal

```
procedure PutFloat(Value: Single); stdcall;
```

C++

```
__fastcall __stdcall PutFloat(Single Value);
```

Description

This is PutFloat, a member of class IMarshalOutBuffer.

52.1.4.2.6 IMarshalOutBuffer.PutLong Method

Pascal

```
procedure PutLong(Value: Integer); stdcall;
```

C++

```
__fastcall __stdcall PutLong(int Value);
```

Description

This is PutLong, a member of class IMarshalOutBuffer.

52.1.4.2.7 IMarshalOutBuffer.PutObject Method

Pascal

```
procedure PutObject(const Value: ICorbaObj); stdcall;
```

C++

```
__fastcall __stdcall PutObject(const ICorbaObj Value);
```

Description

This is PutObject, a member of class IMarshalOutBuffer.

52.1.4.2.8 IMarshalOutBuffer.PutShort Method

Pascal

```
procedure PutShort(Value: Smallint); stdcall;
```

C++

```
__fastcall __stdcall PutShort(Smallint Value);
```

Description

This is PutShort, a member of class IMarshalOutBuffer.

52.1.4.2.9 IMarshalOutBuffer.PutText Method

Pascal

```
procedure PutText(Value: PChar); stdcall;
```

C++

```
__fastcall __stdcall PutText(const char * Value);
```

Description

This is PutText, a member of class IMarshalOutBuffer.

52.1.4.2.10 IMarshalOutBuffer.PutUnsignedChar Method

Pascal

```
procedure PutUnsignedChar(Value: Byte); stdcall;
```

C++

```
__fastcall __stdcall PutUnsignedChar(Byte Value);
```

Description

This is PutUnsignedChar, a member of class IMarshalOutBuffer.

52.1.4.2.11 IMarshalOutBuffer.PutUnsignedLong Method

Pascal

```
procedure PutUnsignedLong(Value: UINT); stdcall;
```

C++

```
__fastcall __stdcall PutUnsignedLong(UINT Value);
```

Description

This is PutUnsignedLong, a member of class IMarshalOutBuffer.

52.1.4.2.12 IMarshalOutBuffer.PutUnsignedShort Method

Pascal

```
procedure PutUnsignedShort(Value: Word); stdcall;
```

C++

```
__fastcall __stdcall PutUnsignedShort(Word Value);
```

Description

This is PutUnsignedShort, a member of class IMarshalOutBuffer.

52.1.4.2.13 IMarshalOutBuffer.PutWidechar Method

Pascal

```
procedure PutWidechar(Value: Word); stdcall;
```

C++

```
__fastcall __stdcall PutWidechar(Word Value);
```

Description

This is PutWidechar, a member of class IMarshalOutBuffer.

52.1.4.2.14 IMarshalOutBuffer.PutWideText Method

Pascal

```
procedure PutWideText(Value: PWideChar); stdcall;
```

C++

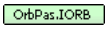
```
__fastcall __stdcall PutWideText(PWideChar Value);
```

Description

This is PutWideText, a member of class IMarshalOutBuffer.

52.1.5 IORB Interface

Class Hierarchy



Pascal

```
IORB = interface;
```

C++

```
__interface IORB;
```

File

OrbPas

Description

This is class OrbPas.IORB.

52.1.5.1 IORB Members

The following tables list the members exposed by IORB.

Public Methods

	Name	Description
	BOAInit (see page 2806)	This is BOAInit, a member of class IORB.
	CreateAliasTC (see page 2806)	This is CreateAliasTC, a member of class IORB.
	CreateArrayTC (see page 2806)	This is CreateArrayTC, a member of class IORB.
	CreateObjRefTC (see page 2807)	This is CreateObjRefTC, a member of class IORB.
	CreateSequenceTC (see page 2807)	This is CreateSequenceTC, a member of class IORB.
	CreateStructTC (see page 2807)	This is CreateStructTC, a member of class IORB.
	CreateTC (see page 2807)	This is CreateTC, a member of class IORB.
	DispatchStruct (see page 2807)	This is DispatchStruct, a member of class IORB.
	FindRepositoryTC (see page 2808)	This is FindRepositoryTC, a member of class IORB.
	MakeAny (see page 2808)	This is MakeAny, a member of class IORB.
	ObjectToString (see page 2808)	This is ObjectToString, a member of class IORB.
	Orb (see page 2808)	This is Orb, a member of class IORB.
	Shutdown (see page 2809)	This is Shutdown, a member of class IORB.
	StringToObject (see page 2809)	This is StringToObject, a member of class IORB.

52.1.5.2 IORB Methods

The methods of the IORB class are listed here.

Public Methods

	Name	Description
	BOAInit (see page 2806)	This is BOAInit, a member of class IORB.

	CreateAliasTC (see page 2806)	This is CreateAliasTC, a member of class IORB.
	CreateArrayTC (see page 2806)	This is CreateArrayTC, a member of class IORB.
	CreateObjRefTC (see page 2807)	This is CreateObjRefTC, a member of class IORB.
	CreateSequenceTC (see page 2807)	This is CreateSequenceTC, a member of class IORB.
	CreateStructTC (see page 2807)	This is CreateStructTC, a member of class IORB.
	CreateTC (see page 2807)	This is CreateTC, a member of class IORB.
	DispatchStruct (see page 2807)	This is DispatchStruct, a member of class IORB.
	FindRepositoryTC (see page 2808)	This is FindRepositoryTC, a member of class IORB.
	MakeAny (see page 2808)	This is MakeAny, a member of class IORB.
	ObjectToString (see page 2808)	This is ObjectToString, a member of class IORB.
	Orb (see page 2808)	This is Orb, a member of class IORB.
	Shutdown (see page 2809)	This is Shutdown, a member of class IORB.
	StringToObject (see page 2809)	This is StringToObject, a member of class IORB.

52.1.5.2.1 IORB.BOAInit Method

Pascal

```
procedure BOAInit(Argc: Integer; const Argv: TArgv; out BOA: IBOA); stdcall;
```

C++

```
__fastcall __stdcall BOAInit(int Argc, const TArgv Argv, IBOA BOA);
```

Description

This is BOAInit, a member of class IORB.

52.1.5.2.2 IORB.CreateAliasTC Method

Pascal

```
procedure CreateAliasTC(RepID: PChar; TypeName: PChar; const TC: ITypeCode; out Result:
ITypeCode); stdcall;
```

C++

```
__fastcall __stdcall CreateAliasTC(const char * RepID, const char * TypeName, const ITypeCode
TC, ITypeCode Result);
```

Description

This is CreateAliasTC, a member of class IORB.

52.1.5.2.3 IORB.CreateArrayTC Method

Pascal

```
procedure CreateArrayTC(Bound: CorbaULong; const TC: ITypeCode; out Result: ITypeCode);
stdcall;
```

C++

```
__fastcall __stdcall CreateArrayTC(CorbaULong Bound, const ITypeCode TC, ITypeCode Result);
```

Description

This is CreateArrayTC, a member of class IORB.

52.1.5.2.4 IORB.CreateObjRefTC Method

Pascal

```
procedure CreateObjRefTC(RepId: PChar; Name: PChar; out Result: ITypeCode); stdcall;
```

C++

```
__fastcall __stdcall CreateObjRefTC(const char * RepId, const char * Name, ITypeCode Result);
```

Description

This is CreateObjRefTC, a member of class IORB.

52.1.5.2.5 IORB.CreateSequenceTC Method

Pascal

```
procedure CreateSequenceTC(Bound: CorbaULong; const TC: ITypeCode; out Result: ITypeCode);
stdcall;
```

C++

```
__fastcall __stdcall CreateSequenceTC(CorbaULong Bound, const ITypeCode TC, ITypeCode Result);
```

Description

This is CreateSequenceTC, a member of class IORB.

52.1.5.2.6 IORB.CreateStructTC Method

Pascal

```
procedure CreateStructTC(Kind: TCKind; RepId: PChar; Name: PChar; Members: TStructMembers;
nMember: Integer; out Result: ITypeCode); stdcall;
```

C++

```
__fastcall __stdcall CreateStructTC(TCKind Kind, const char * RepId, const char * Name,
TStructMembers Members, int nMember, ITypeCode Result);
```

Description

This is CreateStructTC, a member of class IORB.

52.1.5.2.7 IORB.CreateTC Method

Pascal

```
procedure CreateTC(Kind: TCKind; out TC: ITypeCode); stdcall;
```

C++

```
__fastcall __stdcall CreateTC(TCKind Kind, ITypeCode TC);
```

Description

This is CreateTC, a member of class IORB.

52.1.5.2.8 IORB.DispatchStruct Method

Pascal

```
function DispatchStruct(Any: PCorbaAny; CallDesc: PCallDesc; Params: Pointer; out Result:
```

```
  Variant): Integer; stdcall;
```

C++

```
  __fastcall __stdcall int DispatchStruct(PCorbaAny Any, PCallDesc CallDesc, void * Params,
  Variant Result);
```

Description

This is DispatchStruct, a member of class IORB.

52.1.5.2.9 IORB.FindRepositoryTC Method

Pascal

```
  procedure FindRepositoryTC(RepID: PChar; out TC: ITypeCode); stdcall;
```

C++

```
  __fastcall __stdcall FindRepositoryTC(const char * RepID, ITypeCode TC);
```

Description

This is FindRepositoryTC, a member of class IORB.

52.1.5.2.10 IORB.MakeAny Method

Pascal

```
  function MakeAny(const TC: ITypeCode; VS: array of Variant): PCorbaAny; stdcall;
```

C++

```
  __fastcall __stdcall PCorbaAny MakeAny(const ITypeCode TC, array of Variant VS);
```

Description

This is MakeAny, a member of class IORB.

52.1.5.2.11 IORB.ObjectToString Method

Pascal

```
  function ObjectToString(const Obj: IStub): PChar; stdcall;
```

C++

```
  __fastcall __stdcall const char * ObjectToString(const IStub Obj);
```

Description

This is ObjectToString, a member of class IORB.

52.1.5.2.12 IORB.Orb Method

Pascal

```
  function Orb: PCorbaOrb;
```

C++

```
  __fastcall PCorbaOrb Orb();
```

Description

This is Orb, a member of class IORB.

52.1.5.2.13 IORB.Shutdown Method

Pascal

```
procedure Shutdown; stdcall;
```

C++

```
__fastcall __stdcall Shutdown();
```

Description

This is Shutdown, a member of class IORB.

52.1.5.2.14 IORB.StringToObject Method

Pascal

```
procedure StringToObject(Str: PChar; out Obj: IStub); stdcall;
```

C++

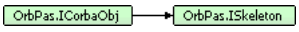
```
__fastcall __stdcall StringToObject(const char * Str, IStub Obj);
```

Description

This is StringToObject, a member of class IORB.

52.1.6 ISkeleton Interface

Class Hierarchy



Pascal

```
ISkeleton = interface(ICorbaObj);
```

C++

```
__interface ISkeleton : public ICorbaObj;
```

File

OrbPas

Description

This is class OrbPas.ISkeleton.

52.1.6.1 ISkeleton Members

The following tables list the members exposed by ISkeleton.

Public Methods

	Name	Description
	CorbaObject (see page 2796)	This is CorbaObject, a member of class ICorbaObj.
	GetImplementation (see page 2810)	This is GetImplementation, a member of class ISkeleton.
	GetReplyBuffer (see page 2810)	This is GetReplyBuffer, a member of class ISkeleton.

	IsLocal (see page 2796)	This is IsLocal, a member of class ICorbaObj.

52.1.6.2 ISkeleton Methods

The methods of the ISkeleton class are listed here.

Public Methods

	Name	Description
	GetImplementation (see page 2810)	This is GetImplementation, a member of class ISkeleton.
	GetReplyBuffer (see page 2810)	This is GetReplyBuffer, a member of class ISkeleton.

52.1.6.2.1 ISkeleton.GetImplementation Method

Pascal

```
procedure GetImplementation(out Impl: IUnknown); stdcall;
```

C++

```
__fastcall __stdcall GetImplementation(IUnknown Impl);
```

Description

This is GetImplementation, a member of class ISkeleton.

52.1.6.2.2 ISkeleton.GetReplyBuffer Method

Pascal

```
procedure GetReplyBuffer(cookie: Pointer; out Outbuf: IMarshalOutBuffer); stdcall;
```

C++

```
__fastcall __stdcall GetReplyBuffer(void * cookie, IMarshalOutBuffer Outbuf);
```

Description

This is GetReplyBuffer, a member of class ISkeleton.

52.1.7 ISkeletonObject Interface

Class Hierarchy

OrbPas.ISkeletonObject

Pascal

```
ISkeletonObject = interface;
```

C++

```
__interface ISkeletonObject;
```

File

OrbPas

Description

This is class OrbPas.ISkeletonObject.

52.1.7.1 ISkeletonObject Members

The following tables list the members exposed by ISkeletonObject.

Public Methods

	Name	Description
	Execute (see page 2811)	This is Execute, a member of class ISkeletonObject.
	GetImplementation (see page 2811)	This is GetImplementation, a member of class ISkeletonObject.
	GetSkeleton (see page 2811)	This is GetSkeleton, a member of class ISkeletonObject.

52.1.7.2 ISkeletonObject Methods

The methods of the ISkeletonObject class are listed here.

Public Methods

	Name	Description
	Execute (see page 2811)	This is Execute, a member of class ISkeletonObject.
	GetImplementation (see page 2811)	This is GetImplementation, a member of class ISkeletonObject.
	GetSkeleton (see page 2811)	This is GetSkeleton, a member of class ISkeletonObject.

52.1.7.2.1 ISkeletonObject.Execute Method

Pascal

```
function Execute(Operation: PAnsiChar; const Strm: IMarshalInBuffer; Cookie: Pointer):
CorbaBoolean; stdcall;
```

C++

```
__fastcall __stdcall CorbaBoolean Execute(PAnsiChar Operation, const IMarshalInBuffer Strm,
void * Cookie);
```

Description

This is Execute, a member of class ISkeletonObject.

52.1.7.2.2 ISkeletonObject.GetImplementation Method

Pascal

```
procedure GetImplementation(out Impl: IUnknown); stdcall;
```

C++

```
__fastcall __stdcall GetImplementation(IUnknown Impl);
```

Description

This is GetImplementation, a member of class ISkeletonObject.

52.1.7.2.3 ISkeletonObject.GetSkeleton Method

Pascal

```
procedure GetSkeleton(out Skeleton: ISkeleton); stdcall;
```

C++

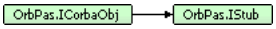
```
__fastcall __stdcall GetSkeleton(ISkeleton Skeleton);
```

Description

This is GetSkeleton, a member of class ISkeletonObject.

52.1.8 IStub Interface

Class Hierarchy



Pascal

```
IStub = interface(ICorbaObj);
```

C++

```
__interface IStub : public ICorbaObj;
```

File

OrbPas

Description

This is class OrbPas.IStub.

52.1.8.1 IStub Members

The following tables list the members exposed by IStub.

Public Methods

	Name	Description
	CorbaObject (see page 2796)	This is CorbaObject, a member of class ICorbaObj.
	CreateRequest (see page 2813)	This is CreateRequest, a member of class IStub.
	Dispatch (see page 2813)	This is Dispatch, a member of class IStub.
	GetInterface (see page 2813)	This is GetInterface, a member of class IStub.
	Hash (see page 2813)	This is Hash, a member of class IStub.
	Invoke (see page 2814)	This is Invoke, a member of class IStub.
	IsA (see page 2814)	This is IsA, a member of class IStub.
	IsLocal (see page 2796)	This is IsLocal, a member of class ICorbaObj.
	NonExistent (see page 2814)	This is NonExistent, a member of class IStub.
	RepositoryID (see page 2814)	This is RepositoryID, a member of class IStub.
	SetPrincipal (see page 2815)	This is SetPrincipal, a member of class IStub.

52.1.8.2 IStub Methods

The methods of the IStub class are listed here.

Public Methods

	Name	Description
	CreateRequest (see page 2813)	This is CreateRequest, a member of class IStub.

	Dispatch (see page 2813)	This is Dispatch, a member of class IStub.
	GetInterface (see page 2813)	This is GetInterface, a member of class IStub.
	Hash (see page 2813)	This is Hash, a member of class IStub.
	Invoke (see page 2814)	This is Invoke, a member of class IStub.
	IsA (see page 2814)	This is IsA, a member of class IStub.
	NonExistent (see page 2814)	This is NonExistent, a member of class IStub.
	RepositoryID (see page 2814)	This is RepositoryID, a member of class IStub.
	SetPrincipal (see page 2815)	This is SetPrincipal, a member of class IStub.

52.1.8.2.1 IStub.CreateRequest Method

Pascal

```
procedure CreateRequest(Operation: PChar; ResponseExpected: CorbaBoolean; out Outbuf:
IMarshalOutBuffer); stdcall;
```

C++

```
__fastcall __stdcall CreateRequest(const char * Operation, CorbaBoolean ResponseExpected,
IMarshalOutBuffer Outbuf);
```

Description

This is CreateRequest, a member of class IStub.

52.1.8.2.2 IStub.Dispatch Method

Pascal

```
function Dispatch(CallDesc: PCallDesc; Params: Pointer; out Result: Variant): Integer; stdcall;
```

C++

```
__fastcall __stdcall int Dispatch(PCallDesc CallDesc, void * Params, Variant Result);
```

Description

This is Dispatch, a member of class IStub.

52.1.8.2.3 IStub.GetInterface Method

Pascal

```
function GetInterface: PCorbaInterfaceDef; stdcall;
```

C++

```
__fastcall __stdcall PCorbaInterfaceDef GetInterface();
```

Description

This is GetInterface, a member of class IStub.

52.1.8.2.4 IStub.Hash Method

Pascal

```
function Hash(Maximum: CorbaULong): CorbaULong; stdcall;
```

C++

```
__fastcall __stdcall CorbaULong Hash(CorbaULong Maximum);
```

Description

This is Hash, a member of class IStub.

52.1.8.2.5 IStub.Invoke Method

Pascal

```
procedure Invoke(const Outbuf: IMarshalOutBuffer; out Inbuf: IMarshalInBuffer); stdcall;
```

C++

```
__fastcall __stdcall Invoke(const IMarshalOutBuffer Outbuf, IMarshalInBuffer Inbuf);
```

Description

This is Invoke, a member of class IStub.

52.1.8.2.6 IStub.IsA Method

Pascal

```
function IsA(LogicalTypeId: PChar): CorbaBoolean; stdcall;
```

C++

```
__fastcall __stdcall CorbaBoolean IsA(const char * LogicalTypeId);
```

Description

This is IsA, a member of class IStub.

52.1.8.2.7 IStub.NonExistent Method

Pascal

```
function NonExistent: CorbaBoolean; stdcall;
```

C++

```
__fastcall __stdcall CorbaBoolean NonExistent();
```

Description

This is NonExistent, a member of class IStub.

52.1.8.2.8 IStub.RepositoryID Method

Pascal

```
function RepositoryID: PChar; stdcall;
```

C++

```
__fastcall __stdcall const char * RepositoryID();
```

Description

This is RepositoryID, a member of class IStub.

52.1.8.2.9 IStub.SetPrincipal Method

Pascal

```
procedure SetPrincipal(Bytes: Pointer; Length: CorbaULong); stdcall;
```

C++

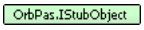
```
__fastcall __stdcall SetPrincipal(void * Bytes, CorbaULong Length);
```

Description

This is SetPrincipal, a member of class IStub.

52.1.9 IStubObject Interface

Class Hierarchy



Pascal

```
IStubObject = interface;
```

C++

```
__interface IStubObject;
```

File

OrbPas

Description

This is class OrbPas.IStubObject.

52.1.9.1 IStubObject Members

The following tables list the members exposed by IStubObject.

Public Methods

	Name	Description
	GetStub (see page 2816)	This is GetStub, a member of class IStubObject.

52.1.9.2 IStubObject Methods

The methods of the IStubObject class are listed here.

Public Methods

	Name	Description
	GetStub (see page 2816)	This is GetStub, a member of class IStubObject.

52.1.9.2.1 IStubObject.GetStub Method

Pascal

```
procedure GetStub(out stub: IStub); stdcall;
```

C++

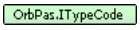
```
__fastcall __stdcall GetStub(IStub stub);
```

Description

This is GetStub, a member of class IStubObject.

52.1.10 ITypeCode Interface

Class Hierarchy



Pascal

```
ITypeCode = interface;
```

C++

```
__interface ITypeCode;
```

File

OrbPas

Description

This is class OrbPas.ITypeCode.

52.1.10.1 ITypeCode Members

The following tables list the members exposed by ITypeCode.

Public Methods

	Name	Description
	ContentType (see page 2817)	This is ContentType, a member of class ITypeCode.
	DefaultIndex (see page 2817)	This is DefaultIndex, a member of class ITypeCode.
	DiscriminatorType (see page 2817)	This is DiscriminatorType, a member of class ITypeCode.
	Equal (see page 2818)	This is Equal, a member of class ITypeCode.
	Id (see page 2818)	This is Id, a member of class ITypeCode.
	Kind (see page 2818)	This is Kind, a member of class ITypeCode.
	Length (see page 2818)	This is Length, a member of class ITypeCode.
	MemberCount (see page 2819)	This is MemberCount, a member of class ITypeCode.
	MemberLabel (see page 2819)	This is MemberLabel, a member of class ITypeCode.
	MemberName (see page 2819)	This is MemberName, a member of class ITypeCode.
	MemberType (see page 2819)	This is MemberType, a member of class ITypeCode.
	Name (see page 2819)	This is Name, a member of class ITypeCode.
	Parameter (see page 2820)	This is Parameter, a member of class ITypeCode.
	ParameterCount (see page 2820)	This is ParameterCount, a member of class ITypeCode.

52.1.10.2 ITypeCode Methods

The methods of the ITypeCode class are listed here.

Public Methods

	Name	Description
	ContentType (see page 2817)	This is ContentType, a member of class ITypeCode.
	DefaultIndex (see page 2817)	This is DefaultIndex, a member of class ITypeCode.
	DiscriminatorType (see page 2817)	This is DiscriminatorType, a member of class ITypeCode.
	Equal (see page 2818)	This is Equal, a member of class ITypeCode.
	Id (see page 2818)	This is Id, a member of class ITypeCode.
	Kind (see page 2818)	This is Kind, a member of class ITypeCode.
	Length (see page 2818)	This is Length, a member of class ITypeCode.
	MemberCount (see page 2819)	This is MemberCount, a member of class ITypeCode.
	MemberLabel (see page 2819)	This is MemberLabel, a member of class ITypeCode.
	MemberName (see page 2819)	This is MemberName, a member of class ITypeCode.
	MemberType (see page 2819)	This is MemberType, a member of class ITypeCode.
	Name (see page 2819)	This is Name, a member of class ITypeCode.
	Parameter (see page 2820)	This is Parameter, a member of class ITypeCode.
	ParameterCount (see page 2820)	This is ParameterCount, a member of class ITypeCode.

52.1.10.2.1 ITypeCode.ContentType Method

Pascal

```
procedure ContentType(out tc: ITypeCode); stdcall;
```

C++

```
__fastcall __stdcall ContentType(ITypeCode tc);
```

Description

This is ContentType, a member of class ITypeCode.

52.1.10.2.2 ITypeCode.DefaultIndex Method

Pascal

```
function DefaultIndex: Integer; stdcall;
```

C++

```
__fastcall __stdcall int DefaultIndex();
```

Description

This is DefaultIndex, a member of class ITypeCode.

52.1.10.2.3 ITypeCode.DiscriminatorType Method

Pascal

```
procedure DiscriminatorType(out tc: ITypeCode); stdcall;
```

C++

```
__fastcall __stdcall DiscriminatorType(ITypeCode tc);
```

Description

This is DiscriminatorType, a member of class ITypeCode.

52.1.10.2.4 ITypeCode.Equal Method

Pascal

```
function Equal(const tc: ITypeCode): CorbaBoolean; stdcall;
```

C++

```
__fastcall __stdcall CorbaBoolean Equal(const ITypeCode tc);
```

Description

This is Equal, a member of class ITypeCode.

52.1.10.2.5 ITypeCode.Id Method

Pascal

```
function Id: PChar; stdcall;
```

C++

```
__fastcall __stdcall const char * Id();
```

Description

This is Id, a member of class ITypeCode.

52.1.10.2.6 ITypeCode.Kind Method

Pascal

```
function Kind: TCKind; stdcall;
```

C++

```
__fastcall __stdcall TCKind Kind();
```

Description

This is Kind, a member of class ITypeCode.

52.1.10.2.7 ITypeCode.Length Method

Pascal

```
function Length: CorbaULong; stdcall;
```

C++

```
__fastcall __stdcall CorbaULong Length();
```

Description

This is Length, a member of class ITypeCode.

52.1.10.2.8 ITypeCode.MemberCount Method

Pascal

```
function MemberCount: CorbaULong; stdcall;
```

C++

```
__fastcall __stdcall CorbaULong MemberCount();
```

Description

This is MemberCount, a member of class ITypeCode.

52.1.10.2.9 ITypeCode.MemberLabel Method

Pascal

```
function MemberLabel(index: CorbaULong): PCorbaAny; stdcall;
```

C++

```
__fastcall __stdcall PCorbaAny MemberLabel(CorbaULong index);
```

Description

This is MemberLabel, a member of class ITypeCode.

52.1.10.2.10 ITypeCode.MemberName Method

Pascal

```
function MemberName(index: CorbaULong): PChar; stdcall;
```

C++

```
__fastcall __stdcall const char * MemberName(CorbaULong index);
```

Description

This is MemberName, a member of class ITypeCode.

52.1.10.2.11 ITypeCode.MemberType Method

Pascal

```
procedure MemberType(index: CorbaULong; out tc: ITypeCode); stdcall;
```

C++

```
__fastcall __stdcall MemberType(CorbaULong index, ITypeCode tc);
```

Description

This is MemberType, a member of class ITypeCode.

52.1.10.2.12 ITypeCode.Name Method

Pascal

```
function Name: PChar; stdcall;
```

C++

```
__fastcall __stdcall const char * Name();
```

Description

This is Name, a member of class ITypeCode.

52.1.10.2.13 ITypeCode.Parameter Method

Pascal

```
function Parameter(index: Integer): PCorbaAny; stdcall;
```

C++

```
__fastcall __stdcall PCorbaAny Parameter(int index);
```

Description

This is Parameter, a member of class ITypeCode.

52.1.10.2.14 ITypeCode.ParameterCount Method

Pascal

```
function ParameterCount: Integer; stdcall;
```

C++

```
__fastcall __stdcall int ParameterCount();
```

Description

This is ParameterCount, a member of class ITypeCode.

52.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	AnyToVariant (see page 2821)	This is function OrbPas.AnyToVariant.
	BindStub (see page 2821)	This is function OrbPas.BindStub.
	CorbaAnyType (see page 2822)	This is function OrbPas.CorbaAnyType.
	CorbaDuplicateAny (see page 2822)	This is function OrbPas.CorbaDuplicateAny.
	CorbaReleaseAny (see page 2822)	This is function OrbPas.CorbaReleaseAny.
	CorbaStringFree (see page 2823)	This is function OrbPas.CorbaStringFree.
	CorbaWStringFree (see page 2823)	This is function OrbPas.CorbaWStringFree.
	CreateSkeleton (see page 2823)	This is function OrbPas.CreateSkeleton.
	CreateUserException (see page 2824)	This is function OrbPas.CreateUserException.
	InitORB (see page 2824)	This is function OrbPas.InitORB.
	RegisterUserException (see page 2824)	This is function OrbPas.RegisterUserException.
	SequenceToVariant (see page 2825)	This is function OrbPas.SequenceToVariant.
	UnRegisterUserException (see page 2825)	This is function OrbPas.UnRegisterUserException.
	VariantToAny (see page 2825)	This is function OrbPas.VariantToAny.

52.2.1 OrbPas.AnyToVariant Function

Pascal

```
function AnyToVariant(Value: PCorbaAny; OV: PVariant): Boolean;
```

C++

```
Boolean AnyToVariant(PCorbaAny Value, PVariant OV);
```

File

OrbPas

Description

This is function OrbPas.AnyToVariant.

52.2.2 OrbPas.BindStub Function

Pascal

```
procedure BindStub(RepositoryID: PChar; InstanceName: PChar; HostName: PChar; const Orb: IORB;
RefCountServer: CorbaBoolean; out Stub: IStub);
```

C++

```
BindStub(const char * RepositoryID, const char * InstanceName, const char * HostName, const
IORB Orb, CorbaBoolean RefCountServer, IStub Stub);
```

File

OrbPas

Description

This is function OrbPas.BindStub.

52.2.3 OrbPas.CorbaAnyType Function

Pascal

```
procedure CorbaAnyType(Any: PCorbaAny; out TypeCode: ITypeCode);
```

C++

```
CorbaAnyType(PCorbaAny Any, ITypeCode TypeCode);
```

File

OrbPas

Description

This is function OrbPas.CorbaAnyType.

52.2.4 OrbPas.CorbaDuplicateAny Function

Pascal

```
function CorbaDuplicateAny(Any: PCorbaAny): PCorbaAny;
```

C++

```
PCorbaAny CorbaDuplicateAny(PCorbaAny Any);
```

File

OrbPas

Description

This is function OrbPas.CorbaDuplicateAny.

52.2.5 OrbPas.CorbaReleaseAny Function

Pascal

```
procedure CorbaReleaseAny(Any: PCorbaAny);
```

C++

```
CorbaReleaseAny(PCorbaAny Any);
```

File

OrbPas

Description

This is function OrbPas.CorbaReleaseAny.

52.2.6 OrbPas.CorbaStringFree Function

Pascal

```
procedure CorbaStringFree(Str: PChar);
```

C++

```
CorbaStringFree(const char * Str);
```

File

OrbPas

Description

This is function OrbPas.CorbaStringFree.

52.2.7 OrbPas.CorbaWStringFree Function

Pascal

```
procedure CorbaWStringFree(Str: PWideChar);
```

C++

```
CorbaWStringFree(PWideChar Str);
```

File

OrbPas

Description

This is function OrbPas.CorbaWStringFree.

52.2.8 OrbPas.CreateSkeleton Function

Pascal

```
procedure CreateSkeleton(InterfaceName: PChar; const DelphiObject: ISkeletonObject; Serialize:
CorbaBoolean; InstanceName: PChar; RepositoryID: PChar; ClientRefCount: CorbaBoolean; out
Skeleton: ISkeleton);
```

C++

```
CreateSkeleton(const char * InterfaceName, const ISkeletonObject DelphiObject, CorbaBoolean
Serialize, const char * InstanceName, const char * RepositoryID, CorbaBoolean ClientRefCount,
ISkeleton Skeleton);
```

File

OrbPas

Description

This is function OrbPas.CreateSkeleton.

52.2.9 OrbPas.CreateUserException Function

Pascal

```
function CreateUserException(copy: TCopyUserExceptionProc; throwobj: TThrowUserExceptionProc):
PUserExceptionProxy;
```

C++

```
PUserExceptionProxy CreateUserException(TCopyUserExceptionProc copy, TThrowUserExceptionProc
throwobj);
```

File

OrbPas

Description

This is function OrbPas.CreateUserException.

52.2.10 OrbPas.InitORB Function

Pascal

```
procedure InitORB(const Argv: TArgv; out Orb: IORB);
```

C++

```
InitORB(const TArgv Argv, IORB Orb);
```

File

OrbPas

Description

This is function OrbPas.InitORB.

52.2.11 OrbPas.RegisterUserException Function

Pascal

```
function RegisterUserException(Name: PChar; RepositoryID: PChar; Factory:
TUserExceptionFactoryProc): PExceptionDescription;
```

C++

```
PExceptionDescription RegisterUserException(const char * Name, const char * RepositoryID,
TUserExceptionFactoryProc Factory);
```

File

OrbPas

Description

This is function OrbPas.RegisterUserException.

52.2.12 OrbPas.SequenceToVariant Function

Pascal

```
function SequenceToVariant(Value: PCorbaAny; OV: PVariant): Boolean;
```

C++

```
Boolean SequenceToVariant(PCorbaAny Value, PVariant OV);
```

File

OrbPas

Description

This is function OrbPas.SequenceToVariant.

52.2.13 OrbPas.UnRegisterUserException Function

Pascal

```
procedure UnRegisterUserException(Description: PExceptionDescription);
```

C++

```
UnRegisterUserException(PExceptionDescription Description);
```

File

OrbPas

Description

This is function OrbPas.UnRegisterUserException.

52.2.14 OrbPas.VariantToAny Function

Pascal

```
function VariantToAny(Value: PVariant): PCorbaAny;
```

C++

```
PCorbaAny VariantToAny(PVariant Value);
```

File

OrbPas

Description

This is function OrbPas.VariantToAny.

52.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TCKind (see page 2826)	This is record OrbPas.TCKind.

Records

	Name	Description
	TStructMember (see page 2827)	This is record OrbPas.TStructMember.

52.3.1 OrbPas.TCKind Enumeration

Pascal

```
TCKind = (
  tk_null,
  tk_void,
  tk_short,
  tk_long,
  tk_ushort,
  tk_ulong,
  tk_float,
  tk_double,
  tk_boolean,
  tk_char,
  tk_octet,
  tk_any,
  tk_TypeCode,
  tk_Principal,
  tk_objref,
  tk_struct,
  tk_union,
  tk_enum,
  tk_string,
  tk_sequence,
  tk_array,
  tk_alias,
  tk_except,
  tk_longlong,
  tk_ulonglong,
  tk_longdouble,
  tk_wchar,
  tk_wstring,
  tk_fixed
);
```

C++

```
enum TCKind {
  tk_null,
  tk_void,
  tk_short,
  tk_long,
  tk_ushort,
```

```
    tk_ulong,
    tk_float,
    tk_double,
    tk_boolean,
    tk_char,
    tk_octet,
    tk_any,
    tk_TypeCode,
    tk_Principal,
    tk_objref,
    tk_struct,
    tk_union,
    tk_enum,
    tk_string,
    tk_sequence,
    tk_array,
    tk_alias,
    tk_except,
    tk_longlong,
    tk_ulonglong,
    tk_longdouble,
    tk_wchar,
    tk_wstring,
    tk_fixed
  };
```

File

OrbPas

Description

This is record OrbPas.TCKind.

52.3.2 OrbPas.TStructMember Record

Pascal

```
  TStructMember = record
    Name: string;
    TC: ITypeCode;
  end;
```

C++

```
  struct TStructMember {
    UnicodeString Name;
    ITypeCode TC;
  };
```

File

OrbPas

Description

This is record OrbPas.TStructMember.

52.4 Types

The following table lists types in this documentation.

Types

Name	Description
CorbaBoolean (see page 2828)	This is type OrbPas.CorbaBoolean.
CorbaULong (see page 2828)	This is type OrbPas.CorbaULong.
PCorbaAny (see page 2829)	This is type OrbPas.PCorbaAny.
PCorbaInterfaceDef (see page 2829)	This is type OrbPas.PCorbaInterfaceDef.
PCorbaObject (see page 2829)	This is type OrbPas.PCorbaObject.
PCorbaOrb (see page 2830)	This is type OrbPas.PCorbaOrb.
PExceptionDescription (see page 2830)	This is type OrbPas.PExceptionDescription.
PMarshalInbuffer (see page 2830)	This is type OrbPas.PMarshalInbuffer.
PMarshalOutbuffer (see page 2831)	This is type OrbPas.PMarshalOutbuffer.
PUserExceptionProxy (see page 2831)	This is type OrbPas.PUserExceptionProxy.
TArgv (see page 2831)	This is type OrbPas.TArgv.
TCopyUserExceptionProc (see page 2832)	This is type OrbPas.TCopyUserExceptionProc.
TStructMembers (see page 2832)	This is type OrbPas.TStructMembers.
TThrowUserExceptionProc (see page 2832)	This is type OrbPas.TThrowUserExceptionProc.
TUserExceptionFactoryProc (see page 2833)	This is type OrbPas.TUserExceptionFactoryProc.

52.4.1 OrbPas.CorbaBoolean Type

Pascal

```
CorbaBoolean = ByteBool;
```

C++

```
ByteBool CorbaBoolean;
```

File

OrbPas

Description

This is type OrbPas.CorbaBoolean.

52.4.2 OrbPas.CorbaULong Type

Pascal

```
CorbaULong = Integer;
```

C++

```
int CorbaULong;
```

File

OrbPas

Description

This is type OrbPas.CorbaULong.

52.4.3 OrbPas.PCorbaAny Type

Pascal

```
PCorbaAny = type Pointer;
```

C++

```
type Pointer PCorbaAny;
```

File

OrbPas

Description

This is type OrbPas.PCorbaAny.

52.4.4 OrbPas.PCorbaInterfaceDef Type

Pascal

```
PCorbaInterfaceDef = type Pointer;
```

C++

```
type Pointer PCorbaInterfaceDef;
```

File

OrbPas

Description

This is type OrbPas.PCorbaInterfaceDef.

52.4.5 OrbPas.PCorbaObject Type

Pascal

```
PCorbaObject = type Pointer;
```

C++

```
type Pointer PCorbaObject;
```

File

OrbPas

Description

This is type OrbPas.PCorbaObject.

52.4.6 OrbPas.PCorbaOrb Type

Pascal

```
PCorbaOrb = type Pointer;
```

C++

```
type Pointer PCorbaOrb;
```

File

OrbPas

Description

This is type OrbPas.PCorbaOrb.

52.4.7 OrbPas.PExceptionDescription Type

Pascal

```
PExceptionDescription = type Pointer;
```

C++

```
type Pointer PExceptionDescription;
```

File

OrbPas

Description

This is type OrbPas.PExceptionDescription.

52.4.8 OrbPas.PMarshalInbuffer Type

Pascal

```
PMarshalInbuffer = type Pointer;
```

C++

```
type Pointer PMarshalInbuffer;
```

File

OrbPas

Description

This is type OrbPas.PMarshalInbuffer.

52.4.9 OrbPas.PMarshalOutbuffer Type

Pascal

```
PMarshalOutbuffer = type Pointer;
```

C++

```
type Pointer PMarshalOutbuffer;
```

File

OrbPas

Description

This is type OrbPas.PMarshalOutbuffer.

52.4.10 OrbPas.PUserExceptionProxy Type

Pascal

```
PUserExceptionProxy = type Pointer;
```

C++

```
type Pointer PUserExceptionProxy;
```

File

OrbPas

Description

This is type OrbPas.PUserExceptionProxy.

52.4.11 OrbPas.TArgv Type

Pascal

```
TArgv = array of string;
```

C++

```
array of string TArgv;
```

File

OrbPas

Description

This is type OrbPas.TArgv.

52.4.12 OrbPas.TCopyUserExceptionProc Type

Pascal

```
TCopyUserExceptionProc = procedure (const InBuf: IMarshalInBuffer) of object;
```

C++

```
(const InBuf: IMarshalInBuffer) ( TCopyUserExceptionProc)();
```

File

OrbPas

Description

This is type OrbPas.TCopyUserExceptionProc.

52.4.13 OrbPas.TStructMembers Type

Pascal

```
TStructMembers = array of TStructMember;
```

C++

```
array of TStructMember TStructMembers;
```

File

OrbPas

Description

This is type OrbPas.TStructMembers.

52.4.14 OrbPas.TThrowUserExceptionProc Type

Pascal

```
TThrowUserExceptionProc = procedure of object;
```

C++

```
( TThrowUserExceptionProc)();
```

File

OrbPas

Description

This is type OrbPas.TThrowUserExceptionProc.

52.4.15 OrbPas.TUserExceptionFactoryProc Type

Pascal

```
TUserExceptionFactoryProc = function : PUserExceptionProxy;
```

C++

```
: PUserExceptionProxy ( TUserExceptionFactoryProc)();
```

File

OrbPas

Description

This is type OrbPas.TUserExceptionFactoryProc.

53 Outline Namespace

This is namespace Outline.

Topics

Name	Description
Classes (see page 2837)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 2957)	The following table lists structs, records, enums in this documentation.
Types (see page 2962)	The following table lists types in this documentation.

Classes

	Name	Description
	EOutlineError (see page 2837)	EOutlineError is the exception class for outline components.
	OutlineError (see page 2838)	OutlineError is a special exception class for invalid outline-node indexes.
	TCustomOutline (see page 2838)	TCustomOutline is the base class for controls that display outline nodes (TOutlineNode (see page 2940) objects).
	TOutline (see page 2874)	TOutline is a multilevel outline that displays the values of TOutlineNode (see page 2940) objects.
	TOutlineNode (see page 2940)	TOutlineNode represents a node in a TCustomOutline (see page 2838) object.

Structs, Records, Enums

	Name	Description
	TAttachMode (see page 2957)	TAttachMode indicates how an outline node should be inserted into an outline.
	TOutlineBitmap (see page 2958)	This is record Outline.TOutlineBitmap.
	TOutlineNodeCompare (see page 2958)	This is record Outline.TOutlineNodeCompare.
	TOutlineOption (see page 2959)	TOutlineOptions (see page 2964) represents the style in which an outline is drawn.
	TOutlineStyle (see page 2959)	TOutlineStyle determines how the nodes of an outline are drawn.
	TOutlineType (see page 2960)	TOutlineType indicates whether an outline is owner-drawn.

Types

Name	Description
EOutlineChange (see page 2962)	EOutlineChange is the type of event handlers that respond when the nodes in an outline control change.
TBitmapArray (see page 2962)	This is type Outline.TBitmapArray.
TBitmapArrayRange (see page 2963)	This is type Outline.TBitmapArrayRange.
TChangeRange (see page 2963)	TChangeRange indicates the direction in which a node is moved.
TOutlineBitmaps (see page 2963)	This is type Outline.TOutlineBitmaps.

TOutlineOptions (see page 2964)	TOutlineOptions represents the style in which an outline is drawn.

53.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	EOutlineError (see page 2837)	EOutlineError is the exception class for outline components.
	OutlineError (see page 2838)	OutlineError is a special exception class for invalid outline-node indexes.
	TCustomOutline (see page 2838)	TCustomOutline is the base class for controls that display outline nodes (TOutlineNode (see page 2940) objects).
	TOutline (see page 2874)	TOutline is a multilevel outline that displays the values of TOutlineNode (see page 2940) objects.
	TOutlineNode (see page 2940)	TOutlineNode represents a node in a TCustomOutline (see page 2838) object.

53.1.1 EOutlineError Class

EOutlineError is the exception class for outline components.

Class Hierarchy

Pascal

```
EOutlineError = class(Exception);
```

C++

```
class EOutlineError : public Exception;
```

File

Outline

Description

EOutlineError is raised when an error occurs as an application works with an outline component. For example, an EOutlineError exception occurs when an application

Attempts to access an outline using an invalid node index.

Attempts to expand a node whose parent isn't expanded.

Attempts to load items from a file using an invalid file extension.

Note: EOutlineError is raised by the TOutline (see page 2874) component, which is maintained for backward compatibility.

See Also

OutlineError (see page 2838)

53.1.2 OutlineError Class

OutlineError is a special exception class for invalid outline-node indexes.

Class Hierarchy

Pascal

```
OutlineError = class(TObject);
```

C++

```
class OutlineError : public TObject;
```

File

Outline

Description

OutlineError is raised when an application cannot access a specified outline node. In most cases, this occurs because the index used is invalid.

OutlineError is raised internally to abort look-ups of outline components. Because it does not descend from Exception, it never appears at runtime—even if the debugger is set to break on exceptions. Thus it is a useful tool for aborting deep recursions.

Note: OutlineError is raised by the TOutline (see page 2874) component, which is maintained for backward compatibility.

See Also

EOutlineError (see page 2837)

Exception

53.1.3 TCustomOutline Class

TCustomOutline is the base class for controls that display outline nodes (TOutlineNode (see page 2940) objects).

Class Hierarchy

Pascal

```
TCustomOutline = class(TCustomGrid);
```

C++

```
class TCustomOutline : public TCustomGrid;
```

File

Outline

Description

Use TCustomOutline as a base class when defining a control that displays outline nodes in a two-dimensional layout. TCustomOutline introduces the ability to manage a hierarchy of outline nodes, including the ability to control the appearance and layout of those nodes and to expand or collapse branches of the outline tree.

Do not create instances of TCustomOutline. To add an outline control to a form, use the TOutline (see page 2874) object, which publishes many of the properties, events, and methods of TCustomOutline, or TTreeView, which wraps the Windows control that performs the same function.

See Also

TOutline (see page 2874)

TOutlineNode (see page 2940)

TTreeView

53.1.3.1 TCustomOutline Members

The following tables list the members exposed by TCustomOutline.

Protected Events

	Name	Description
	OnCollapse (see page 2873)	Occurs immediately after a node in the outline is collapsed.
	OnDrawItem (see page 2873)	Occurs when a node must be drawn in an owner-draw outline.
	OnExpand (see page 2873)	Occurs immediately after a node in the outline is expanded.

Protected Methods

	Name	Description
	BadIndex (see page 2845)	Indicates whether a node's index is up to date.
V	Click (see page 2846)	Updates the SelectedItem (see page 2872) property.
	Collapse (see page 2847)	Generates an OnCollapse (see page 2873) event.
V	DblClick (see page 2847)	Expands or collapses the selected node.
V	DefineProperties (see page 2847)	Creates a binary property for customized bitmap values.
	DeleteNode (see page 2848)	Fixes up the outline after a node has been deleted.
V	DrawCell (see page 2849)	Draws a node in the outline.
	Expand (see page 2850)	Generates an OnExpand (see page 2873) event.
	InvalidateCell (see page 2854)	Invalidates the region occupied by a cell so that it will be repainted with the next Windows paint message.
V	KeyDown (see page 2855)	Provides special processing when the user presses a key.
V	KeyPress (see page 2855)	Provides special processing when a key is pressed.
V	Loaded (see page 2856)	Sets up the outline after it has been streamed in.
V	MouseDown (see page 2858)	Updates the SelectedItem (see page 2872) property.
	Move (see page 2858)	Repositions a node in the outline.
	SetDisplayWidth (see page 2860)	Sets the display width of the outline.
	SetGoodIndex (see page 2860)	Returns the first node in the Items (see page 2865) array that must be reindexed to make the index of the Value parameter correct.
	SetLevel (see page 2861)	Sets the level of an outline node.

Protected Properties

	Name	Description
	ItemHeight (see page 2864)	Specifies the height of each node in the outline.
	ItemSeparator (see page 2865)	Specifies a delimiting string for outline nodes.
	Lines (see page 2866)	Lists the Text property values of all nodes in the outline.
	Options (see page 2866)	Determines how the nodes in the outline are drawn.
	OutlineStyle (see page 2867)	Specifies how the outline tree is displayed.
	PictureClosed (see page 2868)	Specifies the bitmap used to represent a collapsed node that has subitems.
	PictureLeaf (see page 2868)	Specifies the bitmap used to represent an item that does not have subitems.
	PictureMinus (see page 2869)	Specifies the bitmap used to represent an expanded item that has subitems.
	PictureOpen (see page 2869)	Specifies the bitmap used to represent an expanded item that has subitems.
	PicturePlus (see page 2870)	Specifies the bitmap used to represent a collapsed item that has subitems.

	Style (see page 2872)	Determines whether the nodes and tree are drawn automatically.

Public Methods

	Name	Description
	Add (see page 2842)	Adds a node to the outline.
	AddChild (see page 2843)	Adds a child node to a node in the outline.
	AddChildObject (see page 2844)	Adds a child node containing data to a node in the outline.
	AddObject (see page 2844)	Adds a node containing data to the outline.
	BeginUpdate (see page 2846)	Prevents the updating of the outline until the EndUpdate (see page 2849) method is called.
	Clear (see page 2846)	Removes all items from the outline.
V	Create (see page 2840)	Creates an instance of TCustomOutline (see page 2838).
	Delete (see page 2848)	Deletes a node from the outline.
V	Destroy (see page 2841)	Destroys an instance of TCustomOutline (see page 2838).
	EndUpdate (see page 2849)	Performs the reindexing and repaints deferred by a call to BeginUpdate (see page 2846).
	FullCollapse (see page 2850)	Causes all top-level nodes in the outline to collapse.
	FullExpand (see page 2850)	Expands every node in the outline.
	GetDataItem (see page 2851)	Returns the index of the first node in the outline that has the Value pointer as its Data property.
	GetItem (see page 2851)	Returns the index of the node at the position indicated by the X and Y parameters.
	GetNodeDisplayWidth (see page 2852)	Returns the length of the row containing a specified node.
	GetTextItem (see page 2852)	Returns the index of the first node in the outline with the Value parameter as its Text property.
	GetVisibleNode (see page 2852)	Returns the TOutlineNode (see page 2940) object for the node indicated by a position in the image of the outline.
	Insert (see page 2853)	Inserts a node at a specified position.
	InsertObject (see page 2853)	Inserts a node containing data at a specified position.
	LoadFromFile (see page 2856)	This is the overview for the LoadFromFile method overload.
	LoadFromStream (see page 2857)	This is the overview for the LoadFromStream method overload.
	SaveToFile (see page 2858)	This is the overview for the SaveToFile method overload.
	SaveToStream (see page 2859)	This is the overview for the SaveToStream method overload.
	SetUpdateState (see page 2861)	Enables or disables the maintenance of the node indexes.

Public Properties

	Name	Description
	Canvas (see page 2862)	Specifies the TCanvas object that presents a drawing surface for the control.
R	ItemCount (see page 2864)	Indicates the number of nodes in the outline.
R	Items (see page 2865)	Lists the nodes in the outline.
	Row (see page 2870)	Specifies the index of the row that contains the selected cell.
	SelectedItem (see page 2872)	Specifies the index of the selected node.

53.1.3.2 TCustomOutline.Create Constructor

Creates an instance of TCustomOutline (see page 2838).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomOutline(TComponent * AOwner);
```

Description

Call Create to instantiate a TCustomOutline (see page 2838) object at runtime. Outlines placed on forms at design time are created automatically.

Create initializes the outline object, setting the values of some internal properties, as well as setting

Color to clWindow and ParentColor to false.

Height to 97 and Width to 121.

ItemSeparator (see page 2865) to '\'.

Options (see page 2866) to [ooDrawTreeRoot, ooDrawFocusRect].

OutlineStyle (see page 2867) to osTreePictureText.

See Also

Height

ItemSeparator (see page 2865)

Options (see page 2866)

OutlineStyle (see page 2867)

Color

ParentColor

Create

Width

53.1.3.3 TCustomOutline.Destroy Destructor

Destroys an instance of TCustomOutline (see page 2838).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomOutline();
```

Description

Destroy frees all memory associated with the outline. While the destructor frees the TOutlineNode (see page 2940) objects that comprise the outline, any objects associated with those nodes are not freed.

53.1.3.4 TCustomOutline Methods

The methods of the TCustomOutline class are listed here.

Protected Methods

	Name	Description
	BadIndex (see page 2845)	Indicates whether a node's index is up to date.
V	Click (see page 2846)	Updates the SelectedItem (see page 2872) property.
	Collapse (see page 2847)	Generates an OnCollapse (see page 2873) event.
V	DblClick (see page 2847)	Expands or collapses the selected node.
V	DefineProperties (see page 2847)	Creates a binary property for customized bitmap values.
	DeleteNode (see page 2848)	Fixes up the outline after a node has been deleted.
V	DrawCell (see page 2849)	Draws a node in the outline.
	Expand (see page 2850)	Generates an OnExpand (see page 2873) event.

	InvalidateCell (see page 2854)	Invalidates the region occupied by a cell so that it will be repainted with the next Windows paint message.
	KeyDown (see page 2855)	Provides special processing when the user presses a key.
	KeyPress (see page 2855)	Provides special processing when a key is pressed.
	Loaded (see page 2856)	Sets up the outline after it has been streamed in.
	MouseDown (see page 2858)	Updates the SelectedItem (see page 2872) property.
	Move (see page 2858)	Repositions a node in the outline.
	SetDisplayWidth (see page 2860)	Sets the display width of the outline.
	SetGoodIndex (see page 2860)	Returns the first node in the Items (see page 2865) array that must be reindexed to make the index of the Value parameter correct.
	SetLevel (see page 2861)	Sets the level of an outline node.

Public Methods

	Name	Description
	Add (see page 2842)	Adds a node to the outline.
	AddChild (see page 2843)	Adds a child node to a node in the outline.
	AddChildObject (see page 2844)	Adds a child node containing data to a node in the outline.
	AddObject (see page 2844)	Adds a node containing data to the outline.
	BeginUpdate (see page 2846)	Prevents the updating of the outline until the EndUpdate (see page 2849) method is called.
	Clear (see page 2846)	Removes all items from the outline.
	Delete (see page 2848)	Deletes a node from the outline.
	EndUpdate (see page 2849)	Performs the reindexing and repaints deferred by a call to BeginUpdate (see page 2846).
	FullCollapse (see page 2850)	Causes all top-level nodes in the outline to collapse.
	FullExpand (see page 2850)	Expands every node in the outline.
	GetDataItem (see page 2851)	Returns the index of the first node in the outline that has the Value pointer as its Data property.
	GetItem (see page 2851)	Returns the index of the node at the position indicated by the X and Y parameters.
	GetNodeDisplayWidth (see page 2852)	Returns the length of the row containing a specified node.
	GetTextItem (see page 2852)	Returns the index of the first node in the outline with the Value parameter as its Text property.
	GetVisibleNode (see page 2852)	Returns the TOutlineNode (see page 2940) object for the node indicated by a position in the image of the outline.
	Insert (see page 2853)	Inserts a node at a specified position.
	InsertObject (see page 2853)	Inserts a node containing data at a specified position.
	LoadFromFile (see page 2856)	This is the overview for the LoadFromFile method overload.
	LoadFromStream (see page 2857)	This is the overview for the LoadFromStream method overload.
	SaveToFile (see page 2858)	This is the overview for the SaveToFile method overload.
	SaveToStream (see page 2859)	This is the overview for the SaveToStream method overload.
	SetUpdateState (see page 2861)	Enables or disables the maintenance of the node indexes.

53.1.3.4.1 TCustomOutline.Add Method

Adds a node to the outline.

Pascal

```
function Add(Index: LongInt; const Text: string): LongInt;
```

C++

```
__fastcall LongInt Add(LongInt Index, const UnicodeString Text);
```

Description

Use Add to add a TOutlineNode (see page 2940) object to the outline. Text specifies the Text property value of the new node.

Index specifies where to add the new node. The new node is positioned in the outline as the last sibling of the node identified by

Index. To add nodes to the beginning of the outline, specify zero (0) as the Index parameter.

Nodes that appear after the new node are moved down one row and reindexed in the Items (see page 2865) property array with valid Index values. If BeginUpdate (see page 2846) has been called, the Items (see page 2865) array is not reindexed until the EndUpdate (see page 2849) method is called.

Add returns the Index property of the new node.

See Also

AddChild (see page 2843)

AddObject (see page 2844)

BeginUpdate (see page 2846)

Delete (see page 2848)

EndUpdate (see page 2849)

Insert (see page 2853)

Items (see page 2865)

Index (see page 2953)

Text (see page 2955)

53.1.3.4.2 TCustomOutline.AddChild Method

Adds a child node to a node in the outline.

Pascal

```
function AddChild(Index: LongInt; const Text: string): LongInt;
```

C++

```
__fastcall LongInt AddChild(LongInt Index, const UnicodeString Text);
```

Description

Use AddChild to add a TOutlineNode (see page 2940) object as the child of a node in the outline. Text specifies the Text property value of the new node.

Index specifies where to add the new node. The new node is positioned in the outline as the last child of the node identified by Index. To add nodes to the top level of the outline (off the root), specify zero (0) as the Index parameter.

Nodes that appear after the new node are moved down one row and reindexed in the Items (see page 2865) property array with valid Index values. If BeginUpdate (see page 2846) has been called, the Items (see page 2865) array is not reindexed until the EndUpdate (see page 2849) method is called.

AddChild returns the Index property of the new node.

See Also

Add (see page 2842)

AddChildObject (see page 2844)

BeginUpdate (see page 2846)

Delete (see page 2848)

EndUpdate (see page 2849)

Insert (see page 2853)

Items (see page 2865)

Index (see page 2953)

Text (see page 2955)

53.1.3.4.3 TCustomOutline.AddChildObject Method

Adds a child node containing data to a node in the outline.

Pascal

```
function AddChildObject(Index: LongInt; const Text: string; const Data: TCustomData): LongInt;
```

C++

```
__fastcall LongInt AddChildObject(LongInt Index, const UnicodeString Text, const TCustomData
Data);
```

Description

Use AddChildObject to add a TOutlineNode (see page 2940) object as the child of a node in the outline and assign it a data pointer. Text specifies the value of the Text property for the new node. Data specifies the value of the Data property for the new node.

Index specifies where to add the new node. The new node is positioned in the outline as the last child of the node identified by Index. To add nodes to the top level of the outline (off the root), specify zero (0) as the Index parameter.

Nodes that appear after the new node are moved down one row and reindexed in the Items (see page 2865) property array with valid Index values. If BeginUpdate (see page 2846) has been called, the Items (see page 2865) array is not reindexed until the EndUpdate (see page 2849) method is called.

AddChildObject returns the Index property of the new node.

See Also

AddChild (see page 2843)

AddObject (see page 2844)

BeginUpdate (see page 2846)

Delete (see page 2848)

EndUpdate (see page 2849)

InsertObject (see page 2853)

Items (see page 2865)

Data (see page 2951)

Index (see page 2953)

Text (see page 2955)

53.1.3.4.4 TCustomOutline.AddObject Method

Adds a node containing data to the outline.

Pascal

```
function AddObject(Index: LongInt; const Text: string; const Data: TCustomData): LongInt;
```

C++

```
__fastcall LongInt AddObject(LongInt Index, const UnicodeString Text, const TCustomData Data);
```

Description

Use AddObject to add a TOutlineNode (see page 2940) object to the outline and assign it a data pointer. Text specifies the value of the Text property for the new node. Data specifies the value of the Data property for the new node.

Index specifies where to add the new node. The new node is positioned in the outline as the last sibling of the node identified by Index. To add nodes to the beginning of the outline, specify zero (0) as the Index parameter.

Nodes that appear after the new node are moved down one row and reindexed in the Items (see page 2865) property array with valid Index values. If BeginUpdate (see page 2846) has been called, the Items (see page 2865) array is not reindexed until the EndUpdate (see page 2849) method is called.

AddChild (see page 2843) returns the Index property of the new node.

See Also

Add (see page 2842)

AddChildObject (see page 2844)

BeginUpdate (see page 2846)

Delete (see page 2848)

EndUpdate (see page 2849)

InsertObject (see page 2853)

Items (see page 2865)

Data (see page 2951)

Index (see page 2953)

Text (see page 2955)

53.1.3.4.5 TCustomOutline.BadIndex Method

Indicates whether a node's index is up to date.

Pascal

```
function BadIndex(Value: TOutlineNode): Boolean;
```

C++

```
__fastcall Boolean BadIndex(TOutlineNode Value);
```

Description

Use BadIndex to determine whether the Index property of the TOutlineNode (see page 2940) object specified by the Value parameter accurately reflects the position of the node in the outline.

If nodes are added or removed after a call to BeginUpdate (see page 2846), the indexes may be out of date until the EndUpdate (see page 2849) method is called.

See Also

BeginUpdate (see page 2846)

EndUpdate (see page 2849)

Index (see page 2953)

53.1.3.4.6 TCustomOutline.BeginUpdate Method

Prevents the updating of the outline until the EndUpdate (see page 2849) method is called.

Pascal

```
procedure BeginUpdate;
```

C++

```
__fastcall BeginUpdate();
```

Description

Call BeginUpdate before adding, removing, or moving multiple outline items. BeginUpdate speeds processing time by deferring repaints and the adjustment of index values until EndUpdate (see page 2849) is called.

Note: Make sure that EndUpdate (see page 2849) is called after all changes have been made to the nodes in the Items (see page 2865) array.

See Also

BadIndex (see page 2845)

EndUpdate (see page 2849)

Items (see page 2865)

53.1.3.4.7 TCustomOutline.Clear Method

Removes all items from the outline.

Pascal

```
procedure Clear;
```

C++

```
__fastcall Clear();
```

Description

Call Clear to empty the outline of all its nodes. Clear frees the memory for the removed items but not any memory for associated data.

53.1.3.4.8 TCustomOutline.Click Method

Updates the SelectedItem (see page 2872) property.

Pascal

```
procedure Click; override;
```

C++

```
virtual __fastcall Click();
```

Description

Click is called automatically when the user clicks the mouse on a node in the outline. Click updates the SelectedItem (see page 2872) property before the OnClick event handler is called by the inherited method.

See Also

DblClick (see page 2847)

SelectedItem (see page 2872)

53.1.3.4.9 TCustomOutline.Collapse Method

Generates an OnCollapse (see page 2873) event.

Pascal

```
procedure Collapse(Index: LongInt); dynamic;
```

C++

```
__fastcall Collapse(LongInt Index);
```

Description

Collapse is called automatically after a node in the outline is collapsed. It calls the OnCollapse (see page 2873) event handler if it is assigned.

Override the Collapse method to provide additional processing after nodes in the outline have been collapsed, or to block the OnCollapse (see page 2873) event.

See Also

Expand (see page 2850)

OnCollapse (see page 2873)

53.1.3.4.10 TCustomOutline.DblClick Method

Expands or collapses the selected node.

Pascal

```
procedure DblClick; override;
```

C++

```
virtual __fastcall DblClick();
```

Description

DblClick is called automatically when the user double clicks the mouse on a node in the outline. After calling the OnDblClick event handler, DblClick updates the SelectedItem (see page 2872) property and expands the selected node if it is collapsed or collapses the selected node if it is expanded.

See Also

Click (see page 2846)

SelectedItem (see page 2872)

Expanded (see page 2952)

53.1.3.4.11 TCustomOutline.DefineProperties Method

Creates a binary property for customized bitmap values.

Pascal

```
procedure DefineProperties(Filer: TFiler); override;
```

C++

```
virtual __fastcall DefineProperties(TFiler Filer);
```

Description

The streaming system calls DefineProperties to allow the outline to define new properties that should be saved with the outline object. DefineProperties checks whether the outline uses any bitmaps other than the default values of PictureClosed (see page 2868), PictureLeaf (see page 2868), PictureMinus (see page 2869), PictureOpen (see page 2869), and PicturePlus (see page 2870). If the outline has customized any of these bitmaps, DefineProperties uses the DefineBinaryProperty method of the Filer parameter to create a binary property for the customized values.

See Also

PictureClosed (see page 2868)

PictureLeaf (see page 2868)

PictureMinus (see page 2869)

PictureOpen (see page 2869)

PicturePlus (see page 2870)

TWriter

DefineBinaryProperty

53.1.3.4.12 TCustomOutline.Delete Method

Deletes a node from the outline.

Pascal

```
procedure Delete(Index: LongInt);
```

C++

```
__fastcall Delete(LongInt Index);
```

Description

Call Delete to remove a node and all its descendants from the outline. Index is the index of the node to delete.

Delete removes the node and its children, frees all associated memory, and then calls DeleteNode (see page 2848) to update the indexes and make sure that SelectedItem (see page 2872) is valid.

See Also

Add (see page 2842)

DeleteNode (see page 2848)

SelectedItem (see page 2872)

53.1.3.4.13 TCustomOutline.DeleteNode Method

Fixes up the outline after a node has been deleted.

Pascal

```
procedure DeleteNode(Node: TOutlineNode; CurIndex: LongInt);
```

C++

```
__fastcall DeleteNode(TOutlineNode Node, LongInt CurIndex);
```

Description

DeleteNode is called automatically whenever a node in the outline is destroyed. It fixes up the index and makes sure that the

SelectedItem (see page 2872) property is valid.

To Delete (see page 2848) a node from the outline, call the Delete (see page 2848) method of the outline or free the TOulineNode object.

See Also

Delete (see page 2848)

SelectedItem (see page 2872)

53.1.3.4.14 TCustomOutline.DrawCell Method

Draws a node in the outline.

Pascal

```
procedure DrawCell(ACol: Longint; ARow: Longint; ARect: TRect; AState: TGridDrawState);
override;
```

C++

```
virtual __fastcall DrawCell(Longint ACol, Longint ARow, TRect ARect, TGridDrawState AState);
```

Description

DrawCell is called automatically whenever the outline needs to draw a node. If the Style (see page 2872) property is otOwnerDraw and an OnDrawItem (see page 2873) event handler has been supplied, DrawCell generates an OnDrawItem (see page 2873) event. Otherwise, DrawCell draws the node in the manner indicated by the OutlineStyle (see page 2867) property.

See Also

OnDrawItem (see page 2873)

OutlineStyle (see page 2867)

53.1.3.4.15 TCustomOutline.EndUpdate Method

Performs the reindexing and repaints deferred by a call to BeginUpdate (see page 2846).

Pascal

```
procedure EndUpdate;
```

C++

```
__fastcall EndUpdate();
```

Description

Call EndUpdate at the end of a segment of code that began with a call to BeginUpdate (see page 2846). Every time BeginUpdate (see page 2846) is called, an internal counter is incremented. When EndUpdate is called, the counter is decremented. When every call to BeginUpdate (see page 2846) has been matched by a call to EndUpdate, EndUpdate fixes up the indexes and repaints the outline.

See Also

BeginUpdate (see page 2846)

Items (see page 2865)

53.1.3.4.16 TCustomOutline.Expand Method

Generates an OnExpand (see page 2873) event.

Pascal

```
procedure Expand(Index: LongInt); dynamic;
```

C++

```
__fastcall Expand(LongInt Index);
```

Description

Expand is called automatically after a node in the outline is expanded. It calls the OnExpand (see page 2873) event handler if it is assigned.

Override the Expand method to provide additional processing after nodes in the outline have been expanded, or to block the OnExpand (see page 2873) event.

See Also

Collapse (see page 2847)

OnExpand (see page 2873)

53.1.3.4.17 TCustomOutline.FullCollapse Method

Causes all top-level nodes in the outline to collapse.

Pascal

```
procedure FullCollapse;
```

C++

```
__fastcall FullCollapse();
```

Description

Call FullCollapse to shrink the outline as much as possible. FullCollapse sets the Expanded property of each TOutlineNode (see page 2940) that is a direct descendant of the root of the outline to false. This causes all the descendants of these top-level nodes to be hidden. Depending on the value of OutlineStyle (see page 2867), each collapsed top-level node is then represented with the PictureClosed (see page 2868) bitmap and/or the PicturePlus (see page 2870) bitmap.

See Also

OutlineStyle (see page 2867)

PictureClosed (see page 2868)

PicturePlus (see page 2870)

Expanded (see page 2952)

53.1.3.4.18 TCustomOutline.FullExpand Method

Expands every node in the outline.

Pascal

```
procedure FullExpand;
```

C++

```
__fastcall FullExpand();
```

Description

Call FullExpand to show all nodes in the outline. FullExpand calls the FullExpand method of the root node. This causes the Expanded property of all nodes in the outline to be set to true. Every node in the Items (see page 2865) array appears fully expanded in the outline. Depending on the value of OutlineStyle (see page 2867), each node that has descendants is then represented with the PictureOpen (see page 2869) bitmap and/or the PictureMinus (see page 2869) bitmap. The appearance of leaf nodes is unaffected.

See Also

OutlineStyle (see page 2867)

PictureMinus (see page 2869)

PictureOpen (see page 2869)

FullExpand (see page 2945)

53.1.3.4.19 TCustomOutline.GetDataItem Method

Returns the index of the first node in the outline that has the Value pointer as its Data property.

Pascal

```
function GetDataItem(Value: TCustomData): Longint;
```

C++

```
__fastcall Longint GetDataItem(TCustomData Value);
```

Description

Call GetDataItem to locate the node that corresponds to the Value parameter. GetDataItem returns an index into the Items (see page 2865) property array.

See Also

Items (see page 2865)

Data (see page 2951)

53.1.3.4.20 TCustomOutline.GetItem Method

Returns the index of the node at the position indicated by the X and Y parameters.

Pascal

```
function GetItem(X: Integer; Y: Integer): LongInt;
```

C++

```
__fastcall LongInt GetItem(int X, int Y);
```

Description

Call GetItem to locate the node at the specified screen pixel coordinates. GetItem returns an index into the Items (see page 2865) property array.

See Also

Items (see page 2865)

53.1.3.4.21 TCustomOutline.GetNodeDisplayWidth Method

Returns the length of the row containing a specified node.

Pascal

```
function GetNodeDisplayWidth(Node: TOutlineNode): Integer;
```

C++

```
__fastcall int GetNodeDisplayWidth(TOutlineNode Node);
```

Description

Call GetNodeDisplayWidth to determine the length, in pixels, of the line in the outline that contains the node specified by the Node parameter. GetNodeDisplayWidth adjusts for the lines and bitmaps specified by the OutlineStyle (see page 2867) property.

See Also

ItemHeight (see page 2864)

OutlineStyle (see page 2867)

53.1.3.4.22 TCustomOutline.GetTextItem Method

Returns the index of the first node in the outline with the Value parameter as its Text property.

Pascal

```
function GetTextItem(const Value: string): Longint;
```

C++

```
__fastcall Longint GetTextItem(const UnicodeString Value);
```

Description

Call GetTextItem to locate the node that corresponds to the Value parameter. GetTextItem returns an index into the Items (see page 2865) property array.

See Also

Items (see page 2865)

Text (see page 2955)

53.1.3.4.23 TCustomOutline.GetVisibleNode Method

Returns the TOutlineNode (see page 2940) object for the node indicated by a position in the image of the outline.

Pascal

```
function GetVisibleNode(Index: LongInt): TOutlineNode;
```

C++

```
__fastcall TOutlineNode GetVisibleNode(LongInt Index);
```

Description

Call GetVisibleNode to locate a node with the position given by the Index parameter, where all nodes that are hidden because their parent node is collapsed are ignored. An Index of 0 indicates the first visible node, an Index of 1 indicates the second visible node, and so on. If the outline is fully expanded, GetVisibleNode returns the value of Items (see page 2865)[Index+1].

See Also

Items (see page 2865)

53.1.3.4.24 TCustomOutline.Insert Method

Inserts a node at a specified position.

Pascal

```
function Insert(Index: LongInt; const Text: string): LongInt;
```

C++

```
__fastcall LongInt Insert(LongInt Index, const UnicodeString Text);
```

Description

Use Insert to add a TOutlineNode (see page 2940) object to the outline. Text specifies the Text property value of the new node.

Index specifies where to add the new node. The new node is positioned in the outline in the position currently occupied by the node identified by Index. The new node is inserted at the same level as the node currently identified by Index, so that the original node and the new node are siblings and share the same parent. To add nodes to the beginning of the outline, specify zero (0) as the Index parameter.

Nodes that appear after the new node, including the one specified by Index before Insert is called, are moved down one row and reindexed in the Items (see page 2865) property array with valid Index values. If BeginUpdate (see page 2846) has been called, the Items (see page 2865) array is not reindexed until the EndUpdate (see page 2849) method is called.

Insert returns the Index property of the new node.

See Also

Add (see page 2842)

AddChild (see page 2843)

BeginUpdate (see page 2846)

Delete (see page 2848)

EndUpdate (see page 2849)

InsertObject (see page 2853)

Items (see page 2865)

Index (see page 2953)

Text (see page 2955)

53.1.3.4.25 TCustomOutline.InsertObject Method

Inserts a node containing data at a specified position.

Pascal

```
function InsertObject(Index: LongInt; const Text: string; const Data: TCustomData): LongInt;
```

C++

```
__fastcall LongInt InsertObject(LongInt Index, const UnicodeString Text, const TCustomData
Data);
```

Description

Use InsertObject to add a TOutlineNode (see page 2940) object to the outline and assign it a data pointer. Text specifies the value of the Text property for the new node. Data specifies the value of the Data property for the new node.

Index specifies where to add the new node. The new node is positioned in the outline in the position currently occupied by the node identified by Index. The new node is inserted at the same level as the node currently identified by Index, so that the original node and the new node are siblings and share the same parent. To add nodes to the beginning of the outline, specify zero (0) as the Index parameter.

Nodes that appear after the new node, including the one specified by Index before Insert (see page 2853) is called, are moved down one row and reindexed in the Items (see page 2865) property array with valid Index values. If BeginUpdate (see page 2846) has been called, the Items (see page 2865) array is not reindexed until the EndUpdate (see page 2849) method is called.

Insert (see page 2853) returns the Index property of the new node.

See Also

AddObject (see page 2844)

AddChildObject (see page 2844)

BeginUpdate (see page 2846)

Delete (see page 2848)

EndUpdate (see page 2849)

Insert (see page 2853)

Items (see page 2865)

Data (see page 2951)

Index (see page 2953)

Text (see page 2955)

53.1.3.4.26 TCustomOutline.InvalidateCell Method

Invalidates the region occupied by a cell so that it will be repainted with the next Windows paint message.

Pascal

```
procedure InvalidateCell(ACol: Longint; ARow: Longint);
```

C++

```
__fastcall InvalidateCell(Longint ACol, Longint ARow);
```

Description

Call InvalidateCell when the contents of a cell change so that the cell will need to be repainted. The ACol parameter is the column index of the cell, and the ARow parameter is the row index.

See Also

CellRect

InvalidateCol

InvalidateEditor

InvalidateRow

53.1.3.4.27 TCustomOutline.KeyDown Method

Provides special processing when the user presses a key.

Pascal

```
procedure KeyDown(var Key: Word; Shift: TShiftState); override;
```

C++

```
virtual __fastcall KeyDown(Word Key, TShiftState Shift);
```

Description

KeyDown is called automatically when the user presses a key. After calling the inherited method, which generates an OnKeyDown event, KeyDown processes the following keystrokes:

Key	Behavior
Home	Selects the first node in the outline.
End	Selects the last node in the outline.
Return	Toggles the selected node between its collapsed and expanded states.
Ctrl+*	Fully expands the outline.
Right arrow	Selects the first child of the selected node.
Left arrow	Selects the parent of the selected node.
Ctrl+Up arrow	Selects the node following the selected node.
Ctrl+Down arrow	Selects the node preceding the selected node.

See Also

KeyPress (see page 2855)

SelectedItem (see page 2872)

KeyDown

53.1.3.4.28 TCustomOutline.KeyPress Method

Provides special processing when a key is pressed.

Pascal

```
procedure KeyPress(var Key: Char); override;
```

C++

```
virtual __fastcall KeyPress(Char Key);
```

Description

KeyPress is called automatically after the user presses a key but before it is released. After calling the inherited method, which generates an OnKeyPress event, KeyPress processes the following characters :

Key	Behavior
+	Expands the selected node.
-	Collapses the selected node.
*	Fully expands the outline.

See Also

KeyDown (see page 2855)

KeyPress

53.1.3.4.29 TCustomOutline.Loaded Method

Sets up the outline after it has been streamed in.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

The streaming system calls Loaded after the outline has been fully loaded into memory. Loaded makes sure that all indexes are correct, loads any custom bitmaps that replaced the default values, and selects the first node in the outline.

See Also

DefineProperties (see page 2847)

SelectedItem (see page 2872)

SetGoodIndex (see page 2860)

53.1.3.4.30 LoadFromFile Method

This is the overview for the LoadFromFile method overload.

Overload List

	Name	Description
	TCustomOutline.LoadFromFile (string) (see page 2856)	Loads the outline from a text image stored in a file.
	TCustomOutline.LoadFromFile (string, TEncoding) (see page 2856)	Loads the outline from a text image stored in a file.

53.1.3.4.30.1 TCustomOutline.LoadFromFile Method (string)

Loads the outline from a text image stored in a file.

Pascal

```
procedure LoadFromFile(const FileName: string); overload;
procedure LoadFromFile(const FileName: string; Encoding: TEncoding); overload;
```

C++

```
__fastcall LoadFromFile(const UnicodeString FileName);
__fastcall LoadFromFile(const UnicodeString FileName, TEncoding Encoding);
```

Description

Call LoadFromFile to fill an outline from the file specified by the FileName parameter. Each line in the file represents a node in the outline. The file does not include the root node.

Leading tabs and spaces are converted into levels of the outline. Text without any leading tabs or spaces becomes level 1 items.

For example, the following set of strings represents an outline with three first level children descended from the root, and an additional, second level child on the first and third nodes:

```
ChildA
 ChildA1
ChildB
ChildC
 ChildC1
```

To fill the outline from a text image read from a stream, use the LoadFromStream (see page 2857) method. To fill the outline from a text image represented as a TStrings object, use the Lines (see page 2866) property.

See Also

Lines (see page 2866)

LoadFromStream (see page 2857)

SaveToFile (see page 2859)

53.1.3.4.31 LoadFromStream Method

This is the overview for the LoadFromStream method overload.

Overload List

	Name	Description
	TCustomOutline.LoadFromStream (TStream) (see page 2857)	Loads the outline from a text image read from a stream.
	TCustomOutline.LoadFromStream (TStream, TEncoding) (see page 2857)	Loads the outline from a text image read from a stream.

53.1.3.4.31.1 TCustomOutline.LoadFromStream Method (TStream)

Loads the outline from a text image read from a stream.

Pascal

```
procedure LoadFromStream(Stream: TStream); overload;
procedure LoadFromStream(Stream: TStream; Encoding: TEncoding); overload;
```

C++

```
__fastcall LoadFromStream(TStream Stream);
__fastcall LoadFromStream(TStream Stream, TEncoding Encoding);
```

Description

Call LoadFromStream to fill an outline from the stream specified by the Stream parameter. For example, use LoadFromStream to load an outline from an image stored in a database table, using a TBlobStream object. Each node in the outline is separated by a newline character. The stream does not provide the root node.

Leading tabs and spaces are converted into levels of the outline. Text without any leading tabs or spaces becomes level 1 items. For example, the following set of strings represents an outline with three first level children descended from the root, and an additional, second level child on the first and third nodes:

```
ChildA
 ChildA1
ChildB
ChildC
 ChildC1
```

To fill the outline from a text image read from a file, use the LoadFromFile (see page 2856) method. To fill the outline from a

text image represented as a TStrings object, use the Lines (see page 2866) property.

See Also

Lines (see page 2866)

LoadFromFile (see page 2856)

SaveToStream (see page 2859)

53.1.3.4.32 TCustomOutline.MouseDown Method

Updates the SelectedItem (see page 2872) property.

Pascal

```
procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer);
override;
```

C++

```
virtual __fastcall MouseDown(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

After calling the inherited method to generate an OnMouseDown event and locate the line under the mouse, MouseDown sets the SelectedItem (see page 2872) property to the node in the line under the mouse.

See Also

SelectedItem (see page 2872)

53.1.3.4.33 TCustomOutline.Move Method

Repositions a node in the outline.

Pascal

```
procedure Move(Destination: LongInt; Source: LongInt; AttachMode: TAttachMode);
```

C++

```
__fastcall Move(LongInt Destination, LongInt Source, TAttachMode AttachMode);
```

Description

Call Move to reposition the node with the index specified by the Source parameter to the position indicated by the Destination parameter. The meaning of the Destination parameter depends on the value of the AttachMode parameter.

See Also

Add (see page 2842)

AddChild (see page 2843)

Insert (see page 2853)

Items (see page 2865)

53.1.3.4.34 SaveToFile Method

This is the overview for the SaveToFile method overload.

Overload List

	Name	Description
	TCustomOutline.SaveToFile (string) (see page 2859)	Saves the Lines (see page 2866) property to a file.
	TCustomOutline.SaveToFile (string, TEncoding) (see page 2859)	Saves the Lines (see page 2866) property to a file.

53.1.3.4.34.1 TCustomOutline.SaveToFile Method (string)

Saves the Lines (see page 2866) property to a file.

Pascal

```
procedure SaveToFile(const FileName: string); overload;
procedure SaveToFile(const FileName: string; Encoding: TEncoding); overload;
```

C++

```
__fastcall SaveToFile(const UnicodeString FileName);
__fastcall SaveToFile(const UnicodeString FileName, TEncoding Encoding);
```

Description

Call SaveToFile to save a text image of the outline to the file specified by the FileName parameter. The image saved is the value of the Lines (see page 2866) property, where each line of the file represents a node, and each level of the outline is represented by a tab character.

To write the outline to an arbitrary stream, use the SaveToStream (see page 2859) method.

See Also

Lines (see page 2866)

LoadFromFile (see page 2856)

SaveToStream (see page 2859)

53.1.3.4.35 SaveToStream Method

This is the overview for the SaveToStream method overload.

Overload List

	Name	Description
	TCustomOutline.SaveToStream (TStream) (see page 2859)	Writes the Lines (see page 2866) property to a stream.
	TCustomOutline.SaveToStream (TStream, TEncoding) (see page 2859)	Writes the Lines (see page 2866) property to a stream.

53.1.3.4.35.1 TCustomOutline.SaveToStream Method (TStream)

Writes the Lines (see page 2866) property to a stream.

Pascal

```
procedure SaveToStream(Stream: TStream); overload;
procedure SaveToStream(Stream: TStream; Encoding: TEncoding); overload;
```

C++

```
__fastcall SaveToStream(TStream Stream);
__fastcall SaveToStream(TStream Stream, TEncoding Encoding);
```

Description

Call SaveToStream to write a text image of the outline to the stream specified by the Stream parameter. The image saved is the value of the Lines (see page 2866) property, where each line of text represents a node, and each level of the outline is represented by a tab character.

To write the outline to a file, use the SaveToFile (see page 2859) method.

See Also

Lines (see page 2866)

LoadFromStream (see page 2857)

SaveToFile (see page 2859)

53.1.3.4.36 TCustomOutline.SetDisplayWidth Method

Sets the display width of the outline.

Pascal

```
procedure SetDisplayWidth(Value: Integer);
```

C++

```
__fastcall SetDisplayWidth(int Value);
```

Description

Call SetDisplayWidth to set the width of the image of the outline to the number of pixels specified by the Value parameter. Value should large enough to display the most deeply nested node. Use the GetNodeDisplayWidth (see page 2852) method to determine the size needed to display the visible nodes.

If the display width is greater than the client width of the control, users can scroll the image to view the entire width.

See Also

ClientWidth

GetNodeDisplayWidth (see page 2852)

GetVisibleNode (see page 2852)

53.1.3.4.37 TCustomOutline.SetGoodIndex Method

Returns the first node in the Items (see page 2865) array that must be reindexed to make the index of the Value parameter correct.

Pascal

```
function SetGoodIndex(Value: TOutlineNode): TOutlineNode;
```

C++

```
__fastcall TOutlineNode SetGoodIndex(TOutlineNode Value);
```

Description

Call SetGoodIndex to determine how much of the Items (see page 2865) array must be reindexed after changing the nodes in the outline. All nodes from the root to the value returned by SetGoodIndex do not need to be reindexed.

See Also

BadIndex (see page 2845)

53.1.3.4.38 TCustomOutline.SetLevel Method

Sets the level of an outline node.

Pascal

```
procedure SetLevel(Node: TOutlineNode; CurLevel: Cardinal; NewLevel: Cardinal);
```

C++

```
__fastcall SetLevel(TOutlineNode Node, unsigned CurLevel, unsigned NewLevel);
```

Description

Call SetLevel to change the nesting depth of the node specified by the Node parameter. CurLevel is the node's current level and NewLevel is the node's new level.

For example, consider the outline specified by the following value of Lines (see page 2866):

```
NodeA
 NodeB
  NodeC
 NodeD
```

Calling SetLevel for NodeD with CurLevel set to 2 and NewLevel set to 4 results in the outline with the following value of Lines (see page 2866):

```
NodeA
 NodeB
  NodeC
   NodeD
```

That is, the parent of NodeD was changed from NodeA to NodeC.

See Also

Lines (see page 2866)

53.1.3.4.39 TCustomOutline.SetUpdateState Method

Enables or disables the maintenance of the node indexes.

Pascal

```
procedure SetUpdateState(Value: Boolean);
```

C++

```
__fastcall SetUpdateState(Boolean Value);
```

Description

Call SetUpdateState with Value set to true to block the recomputation of indexes. Call SetUpdateState with Value set to false to recompute all indexes in the outline and enable the maintenance of indexes with every change to the nodes in the outline.

Unlike calls to BeginUpdate (see page 2846) and EndUpdate (see page 2849), calls to SetUpdateState do not nest: Calling SetUpdateState with Value set to false always recomputes the indexes regardless of the number of previous calls to SetUpdateState with Value set to true.

Note: Calling SetUpdateState takes precedence over any enabling or disabling of index maintenance provided by BeginUpdate (see page 2846) or EndUpdate (see page 2849).

See Also

BeginUpdate (see page 2846)

EndUpdate (see page 2849)

53.1.3.5 TCustomOutline Properties

The properties of the TCustomOutline class are listed here.

Protected Properties

	Name	Description
	ItemHeight (see page 2864)	Specifies the height of each node in the outline.
	ItemSeparator (see page 2865)	Specifies a delimiting string for outline nodes.
	Lines (see page 2866)	Lists the Text property values of all nodes in the outline.
	Options (see page 2866)	Determines how the nodes in the outline are drawn.
	OutlineStyle (see page 2867)	Specifies how the outline tree is displayed.
	PictureClosed (see page 2868)	Specifies the bitmap used to represent a collapsed node that has subitems.
	PictureLeaf (see page 2868)	Specifies the bitmap used to represent an item that does not have subitems.
	PictureMinus (see page 2869)	Specifies the bitmap used to represent an expanded item that has subitems.
	PictureOpen (see page 2869)	Specifies the bitmap used to represent an expanded item that has subitems.
	PicturePlus (see page 2870)	Specifies the bitmap used to represent a collapsed item that has subitems.
	Style (see page 2872)	Determines whether the nodes and tree are drawn automatically.

Public Properties

	Name	Description
	Canvas (see page 2862)	Specifies the TCanvas object that presents a drawing surface for the control.
R	ItemCount (see page 2864)	Indicates the number of nodes in the outline.
R	Items (see page 2865)	Lists the nodes in the outline.
	Row (see page 2870)	Specifies the index of the row that contains the selected cell.
	SelectedItem (see page 2872)	Specifies the index of the selected node.

53.1.3.5.1 TCustomOutline.Canvas Property

Specifies the TCanvas object that presents a drawing surface for the control.

Pascal

```
property Canvas;
```

C++

```
__property Canvas;
```

Description

Use the properties of the TCanvas object to draw or paint on the surface of the control. Canvas encapsulates a Windows device context, providing all the tools and methods needed for drawing and painting.

Canvas is a protected property that is usually redeclared as public in descendants of TCustomControl.

C++ Examples:

```
/*
The following code uses the bitmaps in an image list
component to draw the contents of each cell in a draw grid.
It draws a focus rectangle around the cell that has focus.
goDrawFocusSelect in the DrawGrid Options parameter must be
True to set focus on a cell.  The ImageList Draw method must
```

```
be called after DrawFocusRect. The OnSelectCell event handler
must implemented to return true.
*/
void __fastcall TForm1::DrawGrid1DrawCell(
  TObject *Sender, int ACol, int ARow, TRect &Rect, TGridDrawState State)
{
  long index = ARow * DrawGrid1->ColCount + ACol;
  DrawGrid1->Canvas->Brush->Color = clBackground;
  DrawGrid1->Canvas->FillRect(Rect);
  if (State.Contains(gdFocused))
  {
    DrawGrid1->Canvas->DrawFocusRect(Rect);
    Label1->Caption = "Cell " + IntToStr(int(index)) + " has the focus.";
  }
  ImageList1->Draw(
    DrawGrid1->Canvas, Rect.Left, Rect.Top, index, True);
}

void __fastcall TForm1::DrawGrid1SelectCell(TObject *Sender, int ACol, int ARow,
      bool &CanSelect)
{
  CanSelect= True;
}
/*
The following code uses ClientRect to find and draw a line
from the top left to the bottom right of the current
control.  Select a TControl in the ListBox list.  This
example assumes that the current object is a descendent of
TControl and has a public Canvas property.
*/
{
  TPoint APoint = Point(X, Y);
  int Index = ListBox1->ItemAtPos(APoint, True);
  TStringGrid *myStringGrid;
  TClass ClassRef = ListBox1->Items->Objects[Index]->ClassType();
  if (String(ClassRef->ClassName()) == "TStringGrid")
  {
    myStringGrid = dynamic_cast<TStringGrid *>(ListBox1->Items->Objects[Index]);
    // make sure we are using the TControls canvas and not the from's!
    myStringGrid->Canvas->MoveTo(myStringGrid->ClientRect.Left, myStringGrid->ClientRect.Top);
    myStringGrid->Canvas->LineTo(myStringGrid->ClientRect.Right,
myStringGrid->ClientRect.Bottom);
    myStringGrid = NULL;
  }
}
```

Delphi Examples:

```
{
The following code uses the bitmaps in an image list
component to draw the contents of each cell in a draw grid.
It draws a focus rectangle around the cell that has focus.
goDrawFocusSelect in the DrawGrid Options parameter must be
True to set focus on a cell.  The ImageList Draw method must
be called after DrawFocusRect. The OnSelectCell event handler
must implemented to return true.
}
procedure TForm1.DrawGrid1DrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
var
  index: Integer;
begin
  index := ARow * DrawGrid1.ColCount + ACol;
  DrawGrid1.Canvas.Brush.Color := clWhite;
  DrawGrid1.Canvas.FillRect(Rect);
  if (gdFocused in State) then
  begin
```

```
    DrawGrid1.Canvas.DrawFocusRect(Rect);
    Label1.Caption:= 'Cell ' + InttoStr(index) + ' has the focus.';
  end;
  ImageList1.Draw(DrawGrid1.Canvas,Rect.Left,Rect.Top,index, True);
end;

procedure TForm1.DrawGrid1SelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
begin
  CanSelect:= True;
end;
{
The following code uses ClientRect to find and draw a line
from the top left to the bottom right of the current
control.  Select a TControl in the ListBox list.  This
example assumes that the current object is a descendent of
TControl and has a public Canvas property.
}
  if (ListBox1.Items.Objects[Index] is TStringGrid) then
  begin
    myStringGrid:= (ListBox1.Items.Objects[Index] as TStringGrid);
    with myStringGrid.ClientRect do
    begin // make sure we are using the TControl's canvas and not the form's!
      myStringGrid.Canvas.MoveTo(Left, Top);
      myStringGrid.Canvas.LineTo(Right, Bottom);
    end;
  myStringGrid:= nil;
  end;
```

See Also

Paint

53.1.3.5.2 TCustomOutline.ItemCount Property

Indicates the number of nodes in the outline.

Pascal

```
property ItemCount: LongInt;
```

C++

```
__property LongInt ItemCount;
```

Description

Read ItemCount to determine the number of nodes in the outline. Use ItemCount with the Items (see page 2865) property array to iterate over all nodes in the outline.

Note: Reading ItemCount can be computationally expensive, as it causes all the indexes used by the outline to be updated.

See Also

Items (see page 2865)

53.1.3.5.3 TCustomOutline.ItemHeight Property

Specifies the height of each node in the outline.

Pascal

```
property ItemHeight: Integer;
```

C++

```
__property int ItemHeight;
```

Description

Set ItemHeight to the height, in pixels, of each node in the outline. Changing the ItemHeight property also changes the font used to display the nodes.

See Also

ItemCount (see page 2864)

Items (see page 2865)

53.1.3.5.4 TCustomOutline.Items Property

Lists the nodes in the outline.

Pascal

```
property Items [Index: LongInt]: TOutlineNode;
```

C++

```
__property TOutlineNode Items[LongInt Index];
```

Description

Use Items to access a node in the outline. Each item is a TOutlineNode (see page 2940) object.

Index is a number specifying the node to return, where Items[0] refers to the root, Items[1] is the first (topmost) node, and so on.

Note: Items is the default property of TCustomOutline (see page 2838). This means that the name of the property may be omitted in code. Thus the line

Note: Outline1.Items[1].Text := Edit1.Text;

Note: can also be written

Note: Outline1[1].Text := Edit1.Text;

Note: In C++, Items can be accessed using the [] operator, to achieve an affect similar to the default property in Delphi.

See Also

ItemCount (see page 2864)

ItemHeight (see page 2864)

ItemSeparator (see page 2865)

Lines (see page 2866)

53.1.3.5.5 TCustomOutline.ItemSeparator Property

Specifies a delimiting string for outline nodes.

Pascal

```
property ItemSeparator: string;
```

C++

```
__property UnicodeString ItemSeparator;
```

Description

Set ItemSeparator to the string that should appear between node names in the path description of any individual node in the outline. ItemSeparator is used to implement the FullPath read method of TOutlineNode (see page 2940).

ItemSeparator is initialized to the backslash (\) character when the TCustomOutline (see page 2838) object is created.

See Also

Items (see page 2865)

FullPath (see page 2952)

53.1.3.5.6 TCustomOutline.Lines Property

Lists the Text property values of all nodes in the outline.

Pascal

```
property Lines: TStrings;
```

C++

```
__property TStrings Lines;
```

Description

Read Lines to obtain a text representation of the outline. Set Lines to fill the outline from a set of text strings. Each string in the Lines property represents a node in the outline. Lines does not include the root node.

Leading tabs and spaces are converted into levels of the outline. Text without any leading tabs or spaces becomes level 1 items. For example, the following set of strings represents an outline with three first level children descended from the root, and an additional, second level child on the first and third nodes:

```
ChildA
 ChildA1
ChildB
ChildC
 ChildC1
```

The Lines property is primarily useful for filling an outline with items at design time. While Lines can be used to access the individual nodes at run time, it is much quicker to access an item with the Items (see page 2865) property.

See Also

Items (see page 2865)

Text (see page 2955)

53.1.3.5.7 TCustomOutline.Options Property

Determines how the nodes in the outline are drawn.

Pascal

```
property Options: TOutlineOptions;
```

C++

```
__property TOutlineOptions Options;
```

Description

Set Options to any combination of the following values:

Value	Meaning
ooDrawTreeRoot	The first item (Index value of 1) is connected to the root item by the outline tree. This means that the tree will extend from the top of the outline to all the first level items. Without ooDrawTreeRoot, all first level items appear leftmost in the outline, not connected by the tree.
ooDrawFocusRect	The outline draws a focus rectangle around the selected item.
ooStretchBitmaps	The outline stretches the standard bitmaps (PictureLeaf (see page 2868), PictureOpen (see page 2869), PictureClosed (see page 2868), PicturePlus (see page 2870), PictureMinus (see page 2869)) to fit in the size of the item, determined by the size of the Font of the Text. Without ooStretchBitmaps, the bitmaps will be cropped if larger than the height of the item text, or won't fill up the entire item space if smaller than the text.

See Also

Items (see page 2865)

OutlineStyle (see page 2867)

PictureClosed (see page 2868)

PictureLeaf (see page 2868)

PictureMinus (see page 2869)

PictureOpen (see page 2869)

PicturePlus (see page 2870)

SelectedItem (see page 2872)

53.1.3.5.8 TCustomOutline.OutlineStyle Property

Specifies how the outline tree is displayed.

Pascal

```
property OutlineStyle: TOutlineStyle;
```

C++

```
__property TOutlineStyle OutlineStyle;
```

Description

Set OutlineStyle to one of the following values:

Value	Meaning
osText	Displays node text (specified in Text) only.
osPlusMinusText	Displays plus picture (specified in PicturePlus (see page 2870)), minus picture (specified in PictureMinus (see page 2869)), and node text (specified in Text).
osPictureText	Displays open picture (specified in PictureOpen (see page 2869)), closed picture (specified in PictureClosed (see page 2868)), leaf picture (specified in PictureLeaf (see page 2868)) and node text (specified in Text).

osPlusMinusPictureText	Displays plus picture (specified in PicturePlus (see page 2870)), minus picture (specified in PictureMinus (see page 2869)), open picture (specified in PictureOpen (see page 2869)), closed picture (specified in PictureClosed (see page 2868)), leaf picture (specified in PictureLeaf (see page 2868)), and node text (specified in Text).
osTreeText	Displays lines connecting outline nodes and node text (specified in Text).
osTreePictureText	Displays lines connecting outline nodes, open picture (specified in PictureOpen (see page 2869)), closed picture (specified in PictureClosed (see page 2868)), leaf picture (specified in PictureLeaf (see page 2868)), and node text (specified in Text).

Note: If the Style (see page 2872) property is otOwnerDraw, the value of OutlineStyle is ignored. The connecting lines, node bitmaps, and node text must all be drawn in an OnDrawItem (see page 2873) event handler.

See Also

Options (see page 2866)

PictureClosed (see page 2868)

PictureLeaf (see page 2868)

PictureMinus (see page 2869)

PictureOpen (see page 2869)

PicturePlus (see page 2870)

Style (see page 2872)

Text (see page 2955)

53.1.3.5.9 TCustomOutline.PictureClosed Property

Specifies the bitmap used to represent a collapsed node that has subitems.

Pascal

```
property PictureClosed: TBitmap;
```

C++

```
__property TBitmap PictureClosed;
```

Description

By default, PictureClosed is a closed file folder bitmap. Set PictureClosed to change the bitmap used for collapsed nodes.

Note: The bitmap is not displayed in the outline unless the OutlineStyle (see page 2867) property is set to osPictureText, osPlusMinusPictureText, or osTreePictureText.

See Also

OutlineStyle (see page 2867)

PictureLeaf (see page 2868)

PictureMinus (see page 2869)

PictureOpen (see page 2869)

53.1.3.5.10 TCustomOutline.PictureLeaf Property

Specifies the bitmap used to represent an item that does not have subitems.

Pascal

```
property PictureLeaf: TBitmap;
```

C++

```
__property TBitmap PictureLeaf;
```

Description

By default, PictureLeaf is a document bitmap. Set PictureLeaf to change the bitmap used for leaf nodes.

Note: The bitmap is not displayed in the outline unless the OutlineStyle (see page 2867) property is set to osPictureText, osPlusMinusPictureText, or osTreePictureText.

See Also

OutlineStyle (see page 2867)

PictureClosed (see page 2868)

PictureOpen (see page 2869)

53.1.3.5.11 TCustomOutline.PictureMinus Property

Specifies the bitmap used to represent an expanded item that has subitems.

Pascal

```
property PictureMinus: TBitmap;
```

C++

```
__property TBitmap PictureMinus;
```

Description

By default, PictureMinus is a minus sign (-). Set PictureMinus to change the bitmap used for collapsible nodes.

Note: The bitmap is not displayed in the outline unless the OutlineStyle (see page 2867) property is set to osPlusMinusPictureText or osPlusMinusText.

See Also

OutlineStyle (see page 2867)

PictureOpen (see page 2869)

PicturePlus (see page 2870)

53.1.3.5.12 TCustomOutline.PictureOpen Property

Specifies the bitmap used to represent an expanded item that has subitems.

Pascal

```
property PictureOpen: TBitmap;
```

C++

```
__property TBitmap PictureOpen;
```

Description

By default, the PictureOpen property is an open file folder. Set PictureOpen to change the bitmap used for expanded nodes.

Note: The bitmap is not displayed in the outline unless the OutlineStyle (see page 2867) property is set to osPictureText,

osPlusMinusPictureText, or osTreePictureText.

See Also

OutlineStyle (see page 2867)

PictureClosed (see page 2868)

PictureLeaf (see page 2868)

PicturePlus (see page 2870)

53.1.3.5.13 TCustomOutline.PicturePlus Property

Specifies the bitmap used to represent a collapsed item that has subitems.

Pascal

```
property PicturePlus: TBitmap;
```

C++

```
__property TBitmap PicturePlus;
```

Description

By default, the PicturePlus property is a plus sign (+). Set PicturePlus to change the bitmap used for collapsed nodes.

Note: The bitmap will not be displayed in the outline unless the OutlineStyle (see page 2867) property is set to osPlusMinusPictureText or osPlusMinusText.

See Also

OutlineStyle (see page 2867)

PictureClosed (see page 2868)

PictureMinus (see page 2869)

53.1.3.5.14 TCustomOutline.Row Property

Specifies the index of the row that contains the selected cell.

Pascal

```
property Row;
```

C++

```
__property Row;
```

Description

Use Row at runtime to determine the current row in the grid. Setting Row moves focus to the cell in the current column that is in the new row. The first row has an index of 0, the second row an index of 1, and so on.

The selected cell in the grid can be located by reading the Row property and the Col property to obtain the indexes of its row and column. When focus moves to the grid, the selected cell gets input focus.

C++ Examples:

```
/*
This example displays a string grid, a list box, and a
button on a form. When the application runs, strings are put
in the cells of row 1 of the string grid. When the user
clicks the button, The strings in the current row of the
grid are displayed in the list box.
```

```
  Note that when the grid contains an empty cell, the list box
  has a corresponding empty string.
  */
  void __fastcall TForm1::Button1Click(TObject *Sender)
  {
    ListBox1->Items = StringGrid1->Rows[StringGrid1->Row];
  }

  void __fastcall TForm1::FormCreate(TObject *Sender)
  {

    StringGrid1->Cells[1][0] = "Column 1";
    StringGrid1->Cells[2][0] = "Column 2";
    StringGrid1->Cells[3][0] = "Column 3";
    StringGrid1->Cells[4][0] = "Column 4";
    StringGrid1->Cells[0][1] = "Row 1";
    StringGrid1->Cells[1][1] = "Object";
    StringGrid1->Cells[2][1] = "Pascal";
    StringGrid1->Cells[3][1] = "is";
    StringGrid1->Cells[4][1] = "excellent";
    StringGrid1->Cells[0][2] = "Row 2";
    StringGrid1->Cells[1][2] = "Delphi";
    StringGrid1->Cells[2][2] = "is";
    StringGrid1->Cells[4][2] = "RAD";

  }
```

Delphi Examples:

```
  {
  This example displays a string grid, a list box, and a
  button on a form. When the application runs, strings are put
  in the cells of row 1 of the string grid. When the user
  clicks the button, The strings in the current row of the
  grid are displayed in the list box.
  Note that when the grid contains an empty cell, the list box
  has a corresponding empty string.
  }
  procedure TForm1.Button1Click(Sender: TObject);
  begin
    ListBox1.Items := StringGrid1.Rows[StringGrid1.Row];
  end;
  procedure TForm1.FormCreate(Sender: TObject);
  begin
    with StringGrid1 do
    begin
      Cells[1,0] := 'Column 1';
      Cells[2,0] := 'Column 2';
      Cells[3,0] := 'Column 3';
      Cells[4,0] := 'Column 4';
      Cells[0,1] := 'Row 1';
      Cells[1,1] := 'Object';
      Cells[2,1] := 'Pascal';
      Cells[3,1] := 'is';
      Cells[4,1] := 'excellent';
      Cells[0,2] := 'Row 2';
      Cells[1,2] := 'Delphi';
      Cells[2,2] := 'is';
      Cells[4,2] := 'RAD';
    end;
  end;
```

See Also

Col

53.1.3.5.15 TCustomOutline.SelectedItem Property

Specifies the index of the selected node.

Pascal

```
property SelectedItem: Longint;
```

C++

```
__property Longint SelectedItem;
```

Description

Read SelectedItem to determine the Index of the selected node. If no node is selected, SelectedItem is 0. Set SelectedItem to set focus to a particular node in the outline.

See Also

Items (see page 2865)

53.1.3.5.16 TCustomOutline.Style Property

Determines whether the nodes and tree are drawn automatically.

Pascal

```
property Style: TOutlineType;
```

C++

```
__property TOutlineType Style;
```

Description

By default, Style is osStandard, meaning that the outline displays items in the style determined by the OutlineStyle (see page 2867) property. Set Style to otOwnerDraw to create owner-draw outlines, meaning that items must be drawn on the canvas of the outline by code in an OnDrawItem (see page 2873) event handler.

Owner-draw outlines can display items other than the Text of a node and the standard bitmaps specified in the PictureClosed (see page 2868), PictureOpen (see page 2869), PictureMinus (see page 2869), PicturePlus (see page 2870), and PictureLeaf (see page 2868) properties. Owner-draw outlines require more programming, however, as the application must specify how to render the image for each node in an OnDrawItem (see page 2873) event handler.

See Also

Canvas

OnDrawItem (see page 2873)

OutlineStyle (see page 2867)

53.1.3.6 TCustomOutline Events

The events of the TCustomOutline class are listed here.

Protected Events

	Name	Description
	OnCollapse (see page 2873)	Occurs immediately after a node in the outline is collapsed.
	OnDrawItem (see page 2873)	Occurs when a node must be drawn in an owner-draw outline.
	OnExpand (see page 2873)	Occurs immediately after a node in the outline is expanded.

53.1.3.6.1 TCustomOutline.OnCollapse Event

Occurs immediately after a node in the outline is collapsed.

Pascal

```
property OnCollapse: EOutlineChange;
```

C++

```
__property EOutlineChange OnCollapse;
```

Description

Write an OnCollapse event handler to perform special processing when a node in the outline is collapsed. The event handler supplies the index of the node that collapsed.

See Also

OnExpand (see page 2873)

53.1.3.6.2 TCustomOutline.OnDrawItem Event

Occurs when a node must be drawn in an owner-draw outline.

Pascal

```
property OnDrawItem: TDrawItemEvent;
```

C++

```
__property TDrawItemEvent OnDrawItem;
```

Description

When the Style (see page 2872) property is set to otOwnerDraw, specify an OnDrawItem event handler to render the image of each node in the outline. An OnDrawItem event handler can display items other than the Text of a node and the standard bitmaps specified in the PictureClosed (see page 2868), PictureOpen (see page 2869), PictureMinus (see page 2869), PicturePlus (see page 2870), and PictureLeaf (see page 2868) properties.

Draw the image of the node specified by the Index parameter onto the canvas of the outline.

See Also

Canvas

PictureClosed (see page 2868)

PictureLeaf (see page 2868)

PictureMinus (see page 2869)

PictureOpen (see page 2869)

PicturePlus (see page 2870)

Style (see page 2872)

Text (see page 2955)

53.1.3.6.3 TCustomOutline.OnExpand Event

Occurs immediately after a node in the outline is expanded.

Pascal

```
property OnExpand: EOutlineChange;
```

C++

```
__property EOutlineChange OnExpand;
```

Description

Write an OnExpand event handler to perform special processing when a node in the outline is expanded. The event handler identifies the node that was just expanded.

See Also

OnCollapse (see page 2873)

53.1.4 TOutline Class

TOutline is a multilevel outline that displays the values of TOutlineNode (see page 2940) objects.

Class Hierarchy

Pascal

```
TOutline = class(TCustomOutline);
```

C++

```
class TOutline : public TCustomOutline;
```

File

Outline

Description

Use an outline to visually organize information in a hierarchical tree. Each item in an outline is contained in a TOutlineNode (see page 2940) object. Outline (see page 2835) controls allow an application to manage a hierarchy of outline nodes, including the ability to control the appearance and layout of those nodes and the ability to expand or collapse branches of the outline tree.

TOutline is provided primarily for backward compatibility. New applications should use TTreeView instead.

See Also

TOutlineNode (see page 2940)

TTreeView

53.1.4.1 TOutline Members

The following tables list the members exposed by TOutline.

Protected Methods

	Name	Description
	BadIndex (see page 2845)	Indicates whether a node's index is up to date.
	Click (see page 2846)	Updates the SelectedItem (see page 2872) property.
	Collapse (see page 2847)	Generates an OnCollapse (see page 2873) event.

	DblClick (see page 2847)	Expands or collapses the selected node.
	DefineProperties (see page 2847)	Creates a binary property for customized bitmap values.
	DeleteNode (see page 2848)	Fixes up the outline after a node has been deleted.
	DrawCell (see page 2849)	Draws a node in the outline.
	Expand (see page 2850)	Generates an OnExpand (see page 2873) event.
	InvalidateCell (see page 2854)	Invalidates the region occupied by a cell so that it will be repainted with the next Windows paint message.
	KeyDown (see page 2855)	Provides special processing when the user presses a key.
	KeyPress (see page 2855)	Provides special processing when a key is pressed.
	Loaded (see page 2856)	Sets up the outline after it has been streamed in.
	MouseDown (see page 2858)	Updates the SelectedItem (see page 2872) property.
	Move (see page 2858)	Repositions a node in the outline.
	SetDisplayWidth (see page 2860)	Sets the display width of the outline.
	SetGoodIndex (see page 2860)	Returns the first node in the Items (see page 2865) array that must be reindexed to make the index of the Value parameter correct.
	SetLevel (see page 2861)	Sets the level of an outline node.

Public Methods

	Name	Description
	Add (see page 2842)	Adds a node to the outline.
	AddChild (see page 2843)	Adds a child node to a node in the outline.
	AddChildObject (see page 2844)	Adds a child node containing data to a node in the outline.
	AddObject (see page 2844)	Adds a node containing data to the outline.
	BeginUpdate (see page 2846)	Prevents the updating of the outline until the EndUpdate (see page 2849) method is called.
	Clear (see page 2846)	Removes all items from the outline.
	Create (see page 2840)	Creates an instance of TCustomOutline (see page 2838).
	Delete (see page 2848)	Deletes a node from the outline.
	Destroy (see page 2841)	Destroys an instance of TCustomOutline (see page 2838).
	EndUpdate (see page 2849)	Performs the reindexing and repaints deferred by a call to BeginUpdate (see page 2846).
	FullCollapse (see page 2850)	Causes all top-level nodes in the outline to collapse.
	FullExpand (see page 2850)	Expands every node in the outline.
	GetDataItem (see page 2851)	Returns the index of the first node in the outline that has the Value pointer as its Data property.
	GetItem (see page 2851)	Returns the index of the node at the position indicated by the X and Y parameters.
	GetNodeDisplayWidth (see page 2852)	Returns the length of the row containing a specified node.
	GetTextItem (see page 2852)	Returns the index of the first node in the outline with the Value parameter as its Text property.
	GetVisibleNode (see page 2852)	Returns the TOutlineNode (see page 2940) object for the node indicated by a position in the image of the outline.
	Insert (see page 2853)	Inserts a node at a specified position.
	InsertObject (see page 2853)	Inserts a node containing data at a specified position.
	LoadFromFile (see page 2856)	This is the overview for the LoadFromFile method overload.
	LoadFromStream (see page 2857)	This is the overview for the LoadFromStream method overload.
	SaveToFile (see page 2858)	This is the overview for the SaveToFile method overload.
	SaveToStream (see page 2859)	This is the overview for the SaveToStream method overload.
	SetUpdateState (see page 2861)	Enables or disables the maintenance of the node indexes.

Public Properties

	Name	Description
	Canvas (see page 2862)	Specifies the TCanvas object that presents a drawing surface for the control.
R	ItemCount (see page 2864)	Indicates the number of nodes in the outline.
R	Items (see page 2865)	Lists the nodes in the outline.
	Row (see page 2870)	Specifies the index of the row that contains the selected cell.
	SelectedItem (see page 2872)	Specifies the index of the selected node.

Published Events

	Name	Description
	OnClick (see page 2906)	Occurs when the user clicks the control.
	OnCollapse (see page 2908)	Occurs immediately after a node in the outline is collapsed.
	OnContextPopup (see page 2908)	Occurs when the user right-clicks the control or otherwise invokes the popup menu (such as using the keyboard).
	OnDblClick (see page 2909)	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.
	OnDragDrop (see page 2909)	Occurs when the user drops an object being dragged.
	OnDragOver (see page 2912)	Occurs when the user drags an object over a control.
	OnDrawItem (see page 2916)	Occurs when a node must be drawn in an owner-draw outline.
	OnEndDock (see page 2916)	Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.
	OnEndDrag (see page 2917)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.
	OnEnter (see page 2918)	Occurs when a control receives the input focus.
	OnExit (see page 2920)	Occurs when the input focus shifts away from one control to another.
	OnExpand (see page 2924)	Occurs immediately after a node in the outline is expanded.
	OnKeyDown (see page 2924)	Occurs when a user presses any key while the control has focus.
	OnKeyPress (see page 2928)	Occurs when a key is pressed.
	OnKeyUp (see page 2929)	Occurs when the user releases a key that has been pressed.
	OnMouseActivate (see page 2931)	Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.
	OnMouseDown (see page 2932)	Occurs when the user presses a mouse button with the mouse pointer over a control.
	OnMouseEnter (see page 2934)	Occurs when the user moves the mouse into a control.
	OnMouseLeave (see page 2934)	Occurs when the user moves the mouse outside of a control.
	OnMouseMove (see page 2935)	Occurs when the user moves the mouse pointer while the mouse pointer is over a control.
	OnMouseUp (see page 2937)	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.
	OnStartDock (see page 2939)	Occurs when the user begins to drag a control with a DragKind of dkDock.
	OnStartDrag (see page 2940)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

Published Properties

	Name	Description
	Align (see page 2878)	Determines how the control aligns within its container (parent control).
	BorderStyle (see page 2881)	Determines whether a single line border is drawn around the grid.
	Color (see page 2881)	Specifies the background color of the control.
	Ctl3D (see page 2882)	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
	DoubleBuffered (see page 2883)	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
	DragCursor (see page 2885)	Indicates the image used to represent the mouse pointer when the control is being dragged.
	DragKind (see page 2885)	Specifies whether the control is being dragged normally or for docking.
	DragMode (see page 2885)	Determines how the control initiates drag-and-drop or drag-and-dock operations.
	Enabled (see page 2887)	Controls whether the control responds to mouse, keyboard, and timer events.
	Font (see page 2887)	Controls the attributes of text written on or in the control.
	ItemHeight (see page 2888)	Specifies the height of each node in the outline.
	ItemSeparator (see page 2889)	Specifies a delimiting string for outline nodes.
	Lines (see page 2889)	Lists the Text property values of all nodes in the outline.
	Options (see page 2890)	Determines how the nodes in the outline are drawn.
	OutlineStyle (see page 2890)	Specifies how the outline tree is displayed.
	ParentColor (see page 2891)	Determines where a control looks for its color information.
	ParentCtl3D (see page 2892)	Determines where a component looks to determine if it should appear three dimensional.

	ParentDoubleBuffered (see page 2892)	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
	ParentFont (see page 2892)	Determines where a control looks for its font information.
	ParentShowHint (see page 2893)	Determines where a control looks to find out if its Help Hint should be shown.
	PictureClosed (see page 2893)	Specifies the bitmap used to represent a collapsed node that has subitems.
	PictureLeaf (see page 2894)	Specifies the bitmap used to represent an item that does not have subitems.
	PictureMinus (see page 2894)	Specifies the bitmap used to represent an expanded item that has subitems.
	PictureOpen (see page 2895)	Specifies the bitmap used to represent an expanded item that has subitems.
	PicturePlus (see page 2895)	Specifies the bitmap used to represent a collapsed item that has subitems.
	PopupMenu (see page 2896)	Identifies the pop-up menu associated with the control.
	ScrollBars (see page 2901)	Specifies whether the grid includes horizontal and vertical scroll bars.
	ShowHint (see page 2901)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	Style (see page 2902)	Determines whether the nodes and tree are drawn automatically.
	TabOrder (see page 2903)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 2904)	Determines if the user can tab to a control.
	Visible (see page 2904)	Determines whether the component appears onscreen.

53.1.4.2 TOutline Properties

The properties of the TOutline class are listed here.

Published Properties

	Name	Description
	Align (see page 2878)	Determines how the control aligns within its container (parent control).
	BorderStyle (see page 2881)	Determines whether a single line border is drawn around the grid.
	Color (see page 2881)	Specifies the background color of the control.
	Ctl3D (see page 2882)	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
	DoubleBuffered (see page 2883)	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
	DragCursor (see page 2885)	Indicates the image used to represent the mouse pointer when the control is being dragged.
	DragKind (see page 2885)	Specifies whether the control is being dragged normally or for docking.
	DragMode (see page 2885)	Determines how the control initiates drag-and-drop or drag-and-dock operations.
	Enabled (see page 2887)	Controls whether the control responds to mouse, keyboard, and timer events.
	Font (see page 2887)	Controls the attributes of text written on or in the control.
	ItemHeight (see page 2888)	Specifies the height of each node in the outline.
	ItemSeparator (see page 2889)	Specifies a delimiting string for outline nodes.
	Lines (see page 2889)	Lists the Text property values of all nodes in the outline.
	Options (see page 2890)	Determines how the nodes in the outline are drawn.
	OutlineStyle (see page 2890)	Specifies how the outline tree is displayed.
	ParentColor (see page 2891)	Determines where a control looks for its color information.
	ParentCtl3D (see page 2892)	Determines where a component looks to determine if it should appear three dimensional.
	ParentDoubleBuffered (see page 2892)	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
	ParentFont (see page 2892)	Determines where a control looks for its font information.
	ParentShowHint (see page 2893)	Determines where a control looks to find out if its Help Hint should be shown.
	PictureClosed (see page 2893)	Specifies the bitmap used to represent a collapsed node that has subitems.
	PictureLeaf (see page 2894)	Specifies the bitmap used to represent an item that does not have subitems.
	PictureMinus (see page 2894)	Specifies the bitmap used to represent an expanded item that has subitems.
	PictureOpen (see page 2895)	Specifies the bitmap used to represent an expanded item that has subitems.
	PicturePlus (see page 2895)	Specifies the bitmap used to represent a collapsed item that has subitems.
	PopupMenu (see page 2896)	Identifies the pop-up menu associated with the control.

	ScrollBars (see page 2901)	Specifies whether the grid includes horizontal and vertical scroll bars.
	ShowHint (see page 2901)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	Style (see page 2902)	Determines whether the nodes and tree are drawn automatically.
	TabOrder (see page 2903)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 2904)	Determines if the user can tab to a control.
	Visible (see page 2904)	Determines whether the component appears onscreen.

53.1.4.2.1 TOutline.Align Property

Determines how the control aligns within its container (parent control).

Pascal

```
property Align;
```

C++

```
__property Align;
```

Description

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes. When the parent is resized, an aligned control also resizes so that it continues to span the top, bottom, left, or right edge of the parent.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel.

Tip: If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

Note: To cause a control to maintain a specified relationship with an edge of its parent, but not necessarily lie along one edge of the parent, use the Anchors property instead.

C++ Examples:

```
/*
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
*/

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  TSplitter* ps = new TSplitter(Form1); // These objects will all be cleaned up by the owner
(Form1).
  TFileListBox* pflb = new TFileListBox(Form1); // The parent (Form1) will clean up these
objects.
```

```
  TDirectoryListBox* pdlb = new TDirectoryListBox(Form1);

  // Line up the directory list box on the left of the form.
  pdlb->Parent = Form1;
  pdlb->Align = alLeft;
  pdlb->Width = Form1->ClientWidth/3;

  // Now use the splitter to divide the directory pane from the file pane.
  ps->Parent = Form1;
  ps->Left = pdlb->Left + pdlb->Width + 1; // Move to right of directory list.
  ps->Align = pdlb->Align; // Give same alignment as directory list.

  // Each pane must be at least one quarter of the form's width.
  ps->MinSize = Form1->ClientWidth/4;

  // Finally, create the last pane - a file list box.
  pflb->Parent = Form1;
  pflb->Align = alClient;
  pdlb->FileList = pflb; // Link the file list box to the directory list box.
}
/*
Dynamically create a TProgressBar control and align it to
the bottom of the form.
*/

#include <ComCtrls.hpp>
#include <Controls.hpp>

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TProgressBar *ProgressBar = new TProgressBar(this);  // The owner will clean this up.
  ProgressBar->Parent = this;
  ProgressBar->Align = alBottom;
  TButton *Button1 = new TButton(this);  // The owner will clean this up.
  Button1->Parent = this;
}
/*
This example dynamically creates a Page Control, then a
series of Tab Sheets on the Page Control.
*/
#include <Comctrls.hpp>
TPageControl* ppc;
const int MAXTABS = 3;
TTabSheet* pts[MAXTABS];
const char * ppcTabTitles[] = { "ShortString", "Orders", "Items", "Parts" };
int iTabTitles = sizeof(ppcTabTitles)/sizeof(ppcTabTitles[0]);
const TColor colorPalette[12] = {
  clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia,
  clTeal, clNavy, clMaroon, clLime, clOlive, clPurple};

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    ppc = new TPageControl(this); // The owner (this) will clean this up.
    ppc->Parent = this;
    ppc->Align = alClient;
    for (int i=0;i<iTabTitles;i++)
  {
        pts[i] = new TTabSheet(this); // The owner (this) will clean this up.
        pts[i]->PageControl = ppc;
        pts[i]->Name = AnsiString("pts") + ppcTabTitles[i];
        pts[i]->Caption = ppcTabTitles[i];
        pts[i]->Brush->Color = colorPalette[i];
  }
}
```

Delphi Examples:

```
{
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  Split: TSplitter;
  Files: TFileListBox;
  Dirs: TDirectoryListBox;
begin
  { first add a directory list box to the form }
  Dirs := TDirectoryListBox.Create(Form1);
  { line it up on the left of the form }
  Dirs.Parent := Form1;
  Dirs.Align := alLeft;
  Dirs.Width := Form1.ClientWidth div 3;
{ now add the splitter to divide the directory pane from the file pane }
  Split := TSplitter.Create(Form1);
  Split.Parent := Form1;
  { make sure the splitter is to the right of the directory list box! }
  Split.Left := Dirs.Left + Dirs.Width + 1;
  Split.Align := Dirs.Align; { give it the same alignment as the directory }
  { each pane must be at least one quarter of the form's width }
  Split.MinSize := Form1.ClientWidth div 4;
  { Finally, create the last pane - a file list box }
  Files := TFileListBox.Create(Form1);
  Files.Parent := Form1;
  Files.Align := alClient;
  Dirs.FileList := Files; { link the file list box to the directory list box }
end;
{
Dynamically create a TProgressBar control and align it to
the bottom of the form.
}

var
  I: Integer;

procedure TForm1.Button1Click(Sender: TObject);
var
  ProgressBar: TProgressBar;
begin
  ProgressBar := TProgressBar.Create(Self);
  with ProgressBar do
  begin
    Parent := Self;
    Align := alBottom;
  end;
end;
{
This example dynamically creates a Page Control, then a
series of Tab Sheets on the Page Control.
}
var
  PageControl1: TPageControl;
  pts: array[0..3] of TTabSheet;
const
  TabTitles: array[0..3] of ShortString = ('Customer', 'Orders', 'Items', 'Parts' );

procedure TForm1.FormCreate(Sender: TObject);
```

```
const
  colorpalette: Array[0..11] of TColor = (
    clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia, clTeal,
    clNavy, clMaroon, clLime, clOlive, clPurple);
var
  i: Integer;
begin
  PageControl1 := TPageControl.Create(Self);
  PageControl1.Parent := Self;
  PageControl1.Align := alClient;
  for i := Low(TabTitles) to High(TabTitles) do
  begin
    pts[i]:= TTabSheet.Create(PageControl1);
    with pts[i] do
    begin
      PageControl := PageControl1;
      Name := 'ts' + TabTitles[i];
      Caption := TabTitles[i];
      Brush.Color := colorPalette[i];
   end;
  end;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  i: Integer;
begin
  for i := Low(TabTitles) to High(TabTitles) do
  begin
    pts[i].Free;
  end;
  PageControl1.Free;
end;
```

See Also

Parent

Anchors

53.1.4.2.2 TOutline.BorderStyle Property

Determines whether a single line border is drawn around the grid.

Pascal

```
property BorderStyle;
```

C++

```
__property BorderStyle;
```

Description

Set BorderStyle to bsSingle to add a single line border around the grid's image. Set BorderStyle to bsNone to omit the border.

53.1.4.2.3 TOutline.Color Property

Specifies the background color of the control.

Pascal

```
property Color;
```

C++

```
__property Color;
```

Description

Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to false.

C++ Examples:

```
/*
The following code changes the color of a shape control
using the Color dialog box. The example displays the Color
dialog box when the Button1 button is clicked, allowing the
user to select a color with the dialog box. When a color
value is selected with the dialog box, it is assigned to
the shape control.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  if (ColorDialog1->Execute())
    Shape1->Brush->Color = ColorDialog1->Color;
}
```

Delphi Examples:

```
{
The following code changes the color of a shape control
using the Color dialog box. The example displays the Color
dialog box when the Button1 button is clicked, allowing the
user to select a color with the dialog box. When a color
value is selected with the dialog box, it is assigned to
the shape control.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  if ColorDialog1.Execute then
    Shape1.Brush.Color := ColorDialog1.Color;
end;
```

See Also

ParentColor

53.1.4.2.4 TOutline.Ctl3D Property

Determines whether a control has a three-dimensional (3-D) or two-dimensional look.

Pascal

```
property Ctl3D;
```

C++

```
__property Ctl3D;
```

Description

Ctl3D is provided for backward compatibility. It is not used by 32-bit versions of Windows or NT4.0 and later, with the exception of Windows XP Home, where it is used on occasion.

On earlier platforms, Ctl3D controlled whether the control had a flat or beveled appearance.

See Also

ParentCtl3D

BevelKind

BevelInner

BevelOuter

BorderWidth

53.1.4.2.5 TOutline.DoubleBuffered Property

Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.

Pascal

```
property DoubleBuffered;
```

C++

```
__property DoubleBuffered;
```

Description

When DoubleBuffered is false, the windowed control paints itself directly to the window. When DoubleBuffered is true, the windowed control paints itself to an in-memory bitmap that is then used to paint the window. Double buffering reduces the amount of flicker when the control repaints, but is more memory intensive.

When a windowed control is a dock site and has an associated dock manager, it must be double-buffered.

Note: Some controls, such as TRichEdit, can't paint themselves into a bitmap. For such controls, DoubleBuffered must be set to false.

C++ Examples:

```
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
```

```
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
dropped label.
}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Not');
  ListBox1.Items.Add('In');
  ListBox1.Items.Add('Alphabetical');
  ListBox1.Items.Add('Order');
end;

// This OnDragOver event handler permits the list box to
// accept a dropped label:

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

// This OnDragDrop event handler implements the drop behavior.

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
    with Sender as TListBox do
    begin
      Font := (Source as TLabel).Font;
      Color := (Source as TLabel).Color;
      DoubleBuffered := true;
      Color := clWindow;
    end;
  end;
end;
```

See Also

DockSite

DockManager

PaintWindow

53.1.4.2.6 TOutline.DragCursor Property

Indicates the image used to represent the mouse pointer when the control is being dragged.

Pascal

```
property DragCursor;
```

C++

```
__property DragCursor;
```

Description

Use the DragCursor property to change the cursor image presented when the control is being dragged.

Note: To make a custom cursor available for the DragCursor property, see the Cursor property.

See Also

Cursor

DragKind

DragMode

53.1.4.2.7 TOutline.DragKind Property

Specifies whether the control is being dragged normally or for docking.

Pascal

```
property DragKind;
```

C++

```
__property DragKind;
```

Description

Use DragKind to get or set whether the control participates in drag-and-drop operations, or drag-and-dock operations.

See Also

DragCursor

DragMode

53.1.4.2.8 TOutline.DragMode Property

Determines how the control initiates drag-and-drop or drag-and-dock operations.

Pascal

```
property DragMode;
```

C++

```
__property DragMode;
```

Description

Use DragMode to control when the user can drag the control. Disable the drag-and-drop or drag-and-dock capability at runtime by setting the DragMode property value to dmManual. Enable automatic dragging by setting DragMode to dmAutomatic.

C++ Examples:

```
/*
The following code uses GetNodeAt to add a dragged node as a
child of the node under the mouse when it is dropped.  This
example requires a populated TreeView.  Also, the TTreeView
DragMode property set to dmAutomatic, and the TTreeView
OnDragOver event handler must be implemented to accept the
drop.
*/
void __fastcall TForm1::TreeView1DragDrop(TObject *Sender, TObject *Source,
      int X, int Y)
{
  if (Source != dynamic_cast<TObject *>(TreeView1) || TreeView1->Selected == NULL)
    return;
  THitTests HT = TreeView1->GetHitTestInfoAt(X, Y);
  TNodeAttachMode AttachMode;
  TTreeNode *Item = TreeView1->GetNodeAt(X, Y);
  if (HT.Contains(htOnItem) || HT.Contains(htOnIcon))
    AttachMode = naAddChild;
  else if (HT.Contains(htNowhere))
    AttachMode = naAdd;
  else if (HT.Contains(htOnIndent))
    AttachMode = naInsert;
  else
    return;
  TreeView1->Selected->MoveTo(Item, AttachMode);
}

void __fastcall TForm1::TreeView1DragOver(TObject *Sender, TObject *Source,
      int X, int Y, TDragState State, bool &Accept)
{
  Accept = (Source == dynamic_cast<TObject *>(TreeView1));
}
```

Delphi Examples:

```
{
The following code uses GetNodeAt to add a dragged node as a
child of the node under the mouse when it is dropped.  This
example requires a populated TreeView.  Also, the TTreeView
DragMode property must be set to dmAutomatic, and the TTreeView
OnDragOver event handler must be implemented to accept the
drop.
}
procedure TForm1.TreeView1DragDrop(Sender, Source: TObject; X, Y: Integer);
var
  AnItem: TTreeNode;
  AttachMode: TNodeAttachMode;
  HT: THitTests;
begin
  if TreeView1.Selected = nil then Exit;
  HT := TreeView1.GetHitTestInfoAt(X, Y);
  AnItem := TreeView1.GetNodeAt(X, Y);
  if (HT - [htOnItem, htOnIcon, htNowhere, htOnIndent] <> HT) then
  begin
    if (htOnItem in HT) or (htOnIcon in HT) then
      AttachMode := naAddChild
    else if htNowhere in HT then AttachMode := naAdd
    else if htOnIndent in HT then AttachMode := naInsert;
    TreeView1.Selected.MoveTo(AnItem, AttachMode);
  end;
end;

procedure TForm1.TreeView1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TTreeView;
end;
```

See Also

BeginDrag

DragCursor

EndDrag

OnDragDrop

OnDragOver

OnEndDrag

OnStartDrag

DragKind

Arranging the Order of Persistent Columns

53.1.4.2.9 TOutline.Enabled Property

Controls whether the control responds to mouse, keyboard, and timer events.

Pascal

```
property Enabled;
```

C++

```
__property Enabled;
```

Description

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to False. Some disabled controls appear dimmed (for example: buttons, check boxes, labels), while others (container controls) simply lose their functionality without changing their appearance. If Enabled is set to False, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to True.

Note: This property applies to all TControl descendants.

See Also

Visible

DisableControls

EnableControls

Enabling Mouse

53.1.4.2.10 TOutline.Font Property

Controls the attributes of text written on or in the control.

Pascal

```
property Font;
```

C++

```
__property Font;
```

Description

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

C++ Examples:

```
/*
To use this example, place a TComboBox and TRichEdit or
TMemo on a form.  During form creation, a list of font names
is loaded into the combo box. When the combo box is clicked,
the memo or Rich Edit control font is set to the
corresponding font name in the combo box.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < Screen->Fonts->Count; i++)
   ComboBox1->Items->Add(Screen->Fonts->Strings[i]);
//  ComboBox1->Items = Screen->Fonts;  // Another way to do it.
}

void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
  RichEdit1->Font->Name = ComboBox1->Text;
}
```

Delphi Examples:

```
{
To use this example, place a TComboBox and TRichEdit or
TMemo on a form.  During form creation, a list of font names
is loaded into the combo box. When the combo box is clicked,
the memo or Rich Edit control font is set to the
corresponding font name in the combo box.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  i : Integer;
begin
  for i := 0 to Screen.Fonts.Count - 1 do
   ComboBox1.Items.Add(Screen.Fonts.Strings[i]);
//  ComboBox1.Items := Screen.Fonts;  // Another way to do it.
  RichEdit1.Lines.Text := 'How is it to be you?';
end;

procedure TForm1.ComboBox1Click(Sender: TObject);
begin
  RichEdit1.Font.Name := ComboBox1.Text;
end;
```

See Also

DesktopFont

ParentFont

Displaying and Editing Text in a Memo Control

53.1.4.2.11 TOutline.ItemHeight Property

Specifies the height of each node in the outline.

Pascal

```
property ItemHeight: Integer;
```

C++

```
__property int ItemHeight;
```

Description

Set ItemHeight to the height, in pixels, of each node in the outline. Changing the ItemHeight property also changes the font used to display the nodes.

See Also

ItemCount (see page 2864)

Items (see page 2865)

53.1.4.2.12 TOutline.ItemSeparator Property

Specifies a delimiting string for outline nodes.

Pascal

```
property ItemSeparator: string;
```

C++

```
__property UnicodeString ItemSeparator;
```

Description

Set ItemSeparator to the string that should appear between node names in the path description of any individual node in the outline. ItemSeparator is used to implement the FullPath read method of TOutlineNode.

ItemSeparator is initialized to the backslash (\) character when the TCustomOutline object is created.

See Also

Items (see page 2865)

FullPath (see page 2952)

53.1.4.2.13 TOutline.Lines Property

Lists the Text property values of all nodes in the outline.

Pascal

```
property Lines: TStrings;
```

C++

```
__property TStrings Lines;
```

Description

Read Lines to obtain a text representation of the outline. Set Lines to fill the outline from a set of text strings. Each string in the Lines property represents a node in the outline. Lines does not include the root node.

Leading tabs and spaces are converted into levels of the outline. Text without any leading tabs or spaces becomes level 1 items. For example, the following set of strings represents an outline with three first level children descended from the root, and an additional, second level child on the first and third nodes:

```
ChildA
 ChildA1
ChildB
ChildC
 ChildC1
```

The Lines property is primarily useful for filling an outline with items at design time. While Lines can be used to access the individual nodes at run time, it is much quicker to access an item with the Items property.

See Also

Items (see page 2865)

Text (see page 2955)

53.1.4.2.14 TOutline.Options Property

Determines how the nodes in the outline are drawn.

Pascal

```
property Options: TOutlineOptions;
```

C++

```
__property TOutlineOptions Options;
```

Description

Set Options to any combination of the following values:

Value	Meaning
ooDrawTreeRoot	The first item (Index value of 1) is connected to the root item by the outline tree. This means that the tree will extend from the top of the outline to all the first level items. Without ooDrawTreeRoot, all first level items appear leftmost in the outline, not connected by the tree.
ooDrawFocusRect	The outline draws a focus rectangle around the selected item.
ooStretchBitmaps	The outline stretches the standard bitmaps (PictureLeaf, PictureOpen, PictureClosed, PicturePlus, PictureMinus) to fit in the size of the item, determined by the size of the Font of the Text. Without ooStretchBitmaps, the bitmaps will be cropped if larger than the height of the item text, or won't fill up the entire item space if smaller than the text.

See Also

Items (see page 2865)

OutlineStyle (see page 2867)

PictureClosed (see page 2868)

PictureLeaf (see page 2868)

PictureMinus (see page 2869)

PictureOpen (see page 2869)

PicturePlus (see page 2870)

SelectedItem (see page 2872)

53.1.4.2.15 TOutline.OutlineStyle Property

Specifies how the outline tree is displayed.

Pascal

```
property OutlineStyle: TOutlineStyle;
```

C++

```
__property TOutlineStyle OutlineStyle;
```

Description

Set OutlineStyle to one of the following values:

Value	Meaning
osText	Displays node text (specified in Text) only.
osPlusMinusText	Displays plus picture (specified in PicturePlus), minus picture (specified in PictureMinus), and node text (specified in Text).
osPictureText	Displays open picture (specified in PictureOpen), closed picture (specified in PictureClosed), leaf picture (specified in PictureLeaf) and node text (specified in Text).
osPlusMinusPictureText	Displays plus picture (specified in PicturePlus), minus picture (specified in PictureMinus), open picture (specified in PictureOpen), closed picture (specified in PictureClosed), leaf picture (specified in PictureLeaf), and node text (specified in Text).
osTreeText	Displays lines connecting outline nodes and node text (specified in Text).
osTreePictureText	Displays lines connecting outline nodes, open picture (specified in PictureOpen), closed picture (specified in PictureClosed), leaf picture (specified in PictureLeaf), and node text (specified in Text).

Note: If the Style property is otOwnerDraw, the value of OutlineStyle is ignored. The connecting lines, node bitmaps, and node text must all be drawn in an OnDrawItem event handler.

See Also

Options (see page 2866)

PictureClosed (see page 2868)

PictureLeaf (see page 2868)

PictureMinus (see page 2869)

PictureOpen (see page 2869)

PicturePlus (see page 2870)

Style (see page 2872)

Text (see page 2955)

53.1.4.2.16 TOutline.ParentColor Property

Determines where a control looks for its color information.

Pascal

```
property ParentColor;
```

C++

```
__property ParentColor;
```

Description

To have a control use the same color as its parent control, set ParentColor to true. If ParentColor is false, the control uses its own Color property.

Set ParentColor to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentColor is true for all controls in a form, changing the background color of the form to gray causes all the controls on the

form to also have a gray background.

When the value of a control's Color property changes, ParentColor becomes false automatically.

See Also

Color

Parent

ParentFont

ParentShowHint

53.1.4.2.17 TOutline.ParentCtl3D Property

Determines where a component looks to determine if it should appear three dimensional.

Pascal

```
property ParentCtl3D;
```

C++

```
__property ParentCtl3D;
```

Description

ParentCtl3D is provided for backwards compatibility. It has no effect on 32-bit versions of Windows or NT 4.0 and later.

ParentCtl3D determines whether the control uses its parent's Ctl3D property.

See Also

Ctl3D

53.1.4.2.18 TOutline.ParentDoubleBuffered Property

ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.

Pascal

```
property ParentDoubleBuffered;
```

C++

```
__property ParentDoubleBuffered;
```

Description

ParentDoubleBuffered is a boolean indicating that the DoubleBuffered property of this component is overridden by the value of the parent's DoubleBuffered property.

53.1.4.2.19 TOutline.ParentFont Property

Determines where a control looks for its font information.

Pascal

```
property ParentFont;
```

C++

```
__property ParentFont;
```

Description

To have a control use the same font as its parent control, set ParentFont to true. If ParentFont is false, the control uses its own Font property. Many controls default ParentFont to true so that all the controls in a form or other container present a uniform appearance. When the value of a control's Font property changes, ParentFont becomes false automatically.

When ParentFont is true for a form, the form uses the default font.

See Also

Font

Parent

ParentColor

ParentShowHint

53.1.4.2.20 TOutline.ParentShowHint Property

Determines where a control looks to find out if its Help Hint should be shown.

Pascal

```
property ParentShowHint;
```

C++

```
__property ParentShowHint;
```

Description

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the control uses the ShowHint property value of its parent. If ParentShowHint is false, the control uses the value of its own ShowHint property.

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note: Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See Also

Hint

Parent

ParentColor

ParentFont

ShowHint

53.1.4.2.21 TOutline.PictureClosed Property

Specifies the bitmap used to represent a collapsed node that has subitems.

Pascal

```
property PictureClosed: TBitmap;
```

C++

```
__property TBitmap PictureClosed;
```

Description

By default, PictureClosed is a closed file folder bitmap. Set PictureClosed to change the bitmap used for collapsed nodes.

Note: The bitmap is not displayed in the outline unless the OutlineStyle property is set to osPictureText, osPlusMinusPictureText, or osTreePictureText.

See Also

OutlineStyle (see page 2867)

PictureLeaf (see page 2868)

PictureMinus (see page 2869)

PictureOpen (see page 2869)

53.1.4.2.22 TOutline.PictureLeaf Property

Specifies the bitmap used to represent an item that does not have subitems.

Pascal

```
property PictureLeaf: TBitmap;
```

C++

```
__property TBitmap PictureLeaf;
```

Description

By default, PictureLeaf is a document bitmap. Set PictureLeaf to change the bitmap used for leaf nodes.

Note: The bitmap is not displayed in the outline unless the OutlineStyle property is set to osPictureText, osPlusMinusPictureText, or osTreePictureText.

See Also

OutlineStyle (see page 2867)

PictureClosed (see page 2868)

PictureOpen (see page 2869)

53.1.4.2.23 TOutline.PictureMinus Property

Specifies the bitmap used to represent an expanded item that has subitems.

Pascal

```
property PictureMinus: TBitmap;
```

C++

```
__property TBitmap PictureMinus;
```

Description

By default, PictureMinus is a minus sign (-). Set PictureMinus to change the bitmap used for collapsible nodes.

Note: The bitmap is not displayed in the outline unless the OutlineStyle property is set to osPlusMinusPictureText or osPlusMinusText.

See Also

OutlineStyle (see page 2867)

PictureOpen (see page 2869)

PicturePlus (see page 2870)

53.1.4.2.24 TOutline.PictureOpen Property

Specifies the bitmap used to represent an expanded item that has subitems.

Pascal

```
property PictureOpen: TBitmap;
```

C++

```
__property TBitmap PictureOpen;
```

Description

By default, the PictureOpen property is an open file folder. Set PictureOpen to change the bitmap used for expanded nodes.

Note: The bitmap is not displayed in the outline unless the OutlineStyle property is set to osPictureText, osPlusMinusPictureText, or osTreePictureText.

See Also

OutlineStyle (see page 2867)

PictureClosed (see page 2868)

PictureLeaf (see page 2868)

PicturePlus (see page 2870)

53.1.4.2.25 TOutline.PicturePlus Property

Specifies the bitmap used to represent a collapsed item that has subitems.

Pascal

```
property PicturePlus: TBitmap;
```

C++

```
__property TBitmap PicturePlus;
```

Description

By default, the PicturePlus property is a plus sign (+). Set PicturePlus to change the bitmap used for collapsed nodes.

Note: The bitmap will not be displayed in the outline unless the OutlineStyle property is set to osPlusMinusPictureText or osPlusMinusText.

See Also

OutlineStyle (see page 2867)

PictureClosed (see page 2868)

PictureMinus (see page 2869)

53.1.4.2.26 TOutline.PopupMenu Property

Identifies the pop-up menu associated with the control.

Pascal

```
property PopupMenu;
```

C++

```
__property PopupMenu;
```

Description

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the control and clicks the right mouse button. If the TPopupMenu's AutoPopup property is true, the pop-up menu appears automatically. If the menu's AutoPopup property is false, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

C++ Examples:

```
/*
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  PopupMenu1->AutoPopup = true;
  Edit1->PopupMenu = PopupMenu1;
  Edit2->PopupMenu = PopupMenu1;
}

/*
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
*/
void __fastcall TForm1::Copy1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CopyToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Cut1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CutToClipboard();
    else
      MessageBeep(0);
  }
```

```
  else
    MessageBeep(0);
}

void __fastcall TForm1::Paste1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->PasteFromClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}
/*
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
    {
        switch (ClickedItem->Tag)
        {
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
```

```
            }
        }
    }
}

/*
To edit or destroy an item, grab its pointer via the Items
property.
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}

void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  PopupMenu1.AutoPopup := True;
  Edit1.PopupMenu := PopupMenu1;
  Edit2.PopupMenu := PopupMenu1;
end;

{
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
}
procedure TForm1.Copy1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
   Edit1.CopyToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
```

```
    Edit2.CopyToClipboard
  else
    Beep;
end;

procedure TForm1.Cut1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.CutToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CutToClipboard
  else
    Beep;
end;

procedure TForm1.Paste1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.PasteFromClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.PasteFromClipboard
  else
    Beep;
end;
{
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
}

type
  TForm1 = class(TForm)
    AddButton: TButton;
    EditButton: TButton;
    DestroyButton: TButton;
    PopupMenu1: TPopupMenu;
    procedure AddButtonClick(Sender: TObject);
    procedure EditButtonClick(Sender: TObject);
    procedure DestroyButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    procedure PopupMenuItemsClick(Sender: TObject);

  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.AddButtonClick(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
    // The owner (PopupMenu1) will cleanup this menu item.
```

```
    NewItem := TMenuItem.Create(PopupMenu1); // create the new item
    index := PopupMenu1.Items.Count;
    PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
    NewItem.Caption := 'Menu Item ' + IntToStr(index);
    NewItem.Tag := index;
    NewItem.OnClick :=
      PopupMenuItemsClick; // assign it an event handler
  end;

  procedure TForm1.PopupMenuItemsClick(Sender: TObject);
  begin
    with Sender as TMenuItem do
    begin
      case Tag of
        0:  ShowMessage('first item clicked');
        1:  ShowMessage('second item clicked');
        2:  ShowMessage('third item clicked');
        3:  ShowMessage('fourth item clicked');
      end;
    end;
  end;

  {
  To edit or destroy an item, grab its pointer via the Items
  property.
  }
  procedure TForm1.EditButtonClick(Sender: TObject);
  var
    ItemToEdit: TMenuItem;
  begin
    ItemToEdit := PopupMenu.Items[1];
    ItemToEdit.Caption := 'Changed Caption';
  end;

  procedure TForm1.DestroyButtonClick(Sender: TObject);
  var
    ItemToDelete: TMenuItem;
  begin
    ItemToDelete := PopupMenu.Items[2];
    ItemToDelete.Free;
  end;

  procedure TForm1.FormCreate(Sender: TObject);
  var
    index: Integer;
    NewItem: TMenuItem;
  begin
    for index := 0 to 3 do
    begin
        // The owner (PopupMenu1) will cleanup this menu item.
      NewItem := TMenuItem.Create(PopupMenu1); // create the new item
      PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
      NewItem.Caption := 'Menu Item ' + IntToStr(index);
      NewItem.Tag := index;
      NewItem.OnClick :=
        PopupMenuItemsClick; // assign it an event handler
    end;
  end;
```

See Also

AutoPopup (see page 1437)

Popup (see page 1434)

OnContextPopup

53.1.4.2.27 TOutline.ScrollBars Property

Specifies whether the grid includes horizontal and vertical scroll bars.

Pascal

```
property ScrollBars;
```

C++

```
__property ScrollBars;
```

Description

Use ScrollBars to give the grid horizontal or vertical scroll bars.

Note: If all the cells in the grid fit in the ClientWidth, no horizontal scroll bar appears, even if ScrollBars is ssHorizontal or ssBoth. If all the cells fit in the ClientHeight, no vertical scroll bar appears, even if ScrollBars is ssVertical or ssBoth.

See Also

ClientHeight

ClientWidth

53.1.4.2.28 TOutline.ShowHint Property

Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.

Pascal

```
property ShowHint;
```

C++

```
__property ShowHint;
```

Description

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either

the control's own ShowHint property must be true, or

the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

C++ Examples:

```
/*
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  Edit1->Hint = "Enter your name";
  Edit1->ShowHint = true;
```

```
    char string[10];
    char index[3];
    for(int i = 1; i <= 10; i++)
    {
      strcpy(string, "Item");
      itoa(i, index, 10);
      strcat(string, index);
      ListBox1->Items->Add(string);
    }
    ListBox1->Hint = "Select an item";
    ListBox1->ShowHint = true;
    Application->ShowHint = true;
}
}
/*
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
*/
```

Delphi Examples:

```
{
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
  Edit1.Hint := 'Enter your name';
  Edit1.ShowHint := True;
  with ListBox1 do
  begin
    for I := 1 to 10 do
      Items.Add('Item ' + IntToStr(I));
    Hint := 'Select an item';
    ShowHint := True;
  end;
  Application.ShowHint := True;
end;
{
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
}
```

See Also

Hint

ParentShowHint

ShowHint

Displaying Fly-over Help

53.1.4.2.29 TOutline.Style Property

Determines whether the nodes and tree are drawn automatically.

Pascal

```
property Style: TOutlineType;
```

C++

```
__property TOutlineType Style;
```

Description

By default, Style is osStandard, meaning that the outline displays items in the style determined by the OutlineStyle property. Set Style to otOwnerDraw to create owner-draw outlines, meaning that items must be drawn on the canvas of the outline by code in an OnDrawItem event handler.

Owner-draw outlines can display items other than the Text of a node and the standard bitmaps specified in the PictureClosed, PictureOpen, PictureMinus, PicturePlus, and PictureLeaf properties. Owner-draw outlines require more programming, however, as the application must specify how to render the image for each node in an OnDrawItem event handler.

See Also

Canvas

OnDrawItem (see page 2873)

OutlineStyle (see page 2867)

53.1.4.2.30 TOutline.TabOrder Property

Indicates the position of the control in its parent's tab order.

Pascal

```
property TabOrder;
```

C++

```
__property TabOrder;
```

Description

TabOrder is the order in which child windows are visited when the user presses the Tab key. The control with the TabOrder value of 0 is the control that has the focus when the form first appears.

Initially, the tab order is always the order in which the controls were added to the form. The first control added to the form has a TabOrder value of 0, the second is 1, the third is 2, and so on. Change this by changing the TabOrder property.

Each control has a unique tab-order value within its parent. If you change the TabOrder property value of one control to be the same as the value of a second control, the TabOrder value for all the other controls changes. For example, suppose a control is sixth in the tab order. If you change the control's TabOrder property value to 3 (making the control fourth in the tab order), the control that was originally fourth in the tab order now becomes fifth, and the control that was fifth becomes sixth.

Assigning TabOrder a value greater than the number of controls contained in the parent control moves the control to the end of the tab order. The control does not take on the assigned value of TabOrder, but instead is given the number that assures the control is the last in the tab order.

Note: TabOrder is meaningful only if the TabStop property is true and if the control has a parent. (The TabOrder property of a form is not used unless the form is the child of another form.) A control with a TabOrder of -1 has no parent, and therefore cannot be reached by pressing the Tab key. To remove a parented control from the Tab order, set its TabStop property to false.

See Also

FindNextControl

TabStop

53.1.4.2.31 TOutline.TabStop Property

Determines if the user can tab to a control.

Pascal

```
property TabStop;
```

C++

```
__property TabStop;
```

Description

Use the TabStop to allow or disallow access to the control using the Tab key.

If TabStop is true, the control is in the tab order. If TabStop is false, the control is not in the tab order and the user can't press the Tab key to move to the control.

Note: TabStop is not meaningful for a form unless the form assigns another form to be its parent.

See Also

TabOrder

53.1.4.2.32 TOutline.Visible Property

Determines whether the component appears onscreen.

Pascal

```
property Visible;
```

C++

```
__property Visible;
```

Description

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

Calling the Show method sets the control's Visible property to true. Calling the Hide method sets it to false.

C++ Examples:

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
```

```
    {
      TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
      pPage->PageControl = PageControl1;
      pPage->Caption = AnsiString("Page") + IntToStr(i);
      TEdit *pEdit = new TEdit(this); // The owner will clean this up.
      pEdit->Parent = pPage;
      pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
      pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
    }
    PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
    begin
      PageControl := PageControl1;
      Caption := 'TabSheet' + IntToStr(i);
      with TEdit.Create(Self) do
      begin
        Parent := PageControl1.Pages[i];
        Left := Random(PageControl1.ActivePage.ClientWidth - Width);
        Top := Random(PageControl1.ActivePage.ClientHeight - Height);
      end;
    end;
    PageControl1Change(Sender);
end;
```

See Also

Enabled

Hide

Show

VisibleChanging

53.1.4.3 TOutline Events

The events of the TOutline class are listed here.

Published Events

	Name	Description
	OnClick (see page 2906)	Occurs when the user clicks the control.
	OnCollapse (see page 2908)	Occurs immediately after a node in the outline is collapsed.
	OnContextPopup (see page 2908)	Occurs when the user right-clicks the control or otherwise invokes the popup menu (such as using the keyboard).
	OnDblClick (see page 2909)	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.
	OnDragDrop (see page 2909)	Occurs when the user drops an object being dragged.
	OnDragOver (see page 2912)	Occurs when the user drags an object over a control.
	OnDrawItem (see page 2916)	Occurs when a node must be drawn in an owner-draw outline.
	OnEndDock (see page 2916)	Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.
	OnEndDrag (see page 2917)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.
	OnEnter (see page 2918)	Occurs when a control receives the input focus.
	OnExit (see page 2920)	Occurs when the input focus shifts away from one control to another.
	OnExpand (see page 2924)	Occurs immediately after a node in the outline is expanded.
	OnKeyDown (see page 2924)	Occurs when a user presses any key while the control has focus.
	OnKeyPress (see page 2928)	Occurs when a key is pressed.
	OnKeyUp (see page 2929)	Occurs when the user releases a key that has been pressed.
	OnMouseActivate (see page 2931)	Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.
	OnMouseDown (see page 2932)	Occurs when the user presses a mouse button with the mouse pointer over a control.
	OnMouseEnter (see page 2934)	Occurs when the user moves the mouse into a control.
	OnMouseLeave (see page 2934)	Occurs when the user moves the mouse outside of a control.
	OnMouseMove (see page 2935)	Occurs when the user moves the mouse pointer while the mouse pointer is over a control.
	OnMouseUp (see page 2937)	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.
	OnStartDock (see page 2939)	Occurs when the user begins to drag a control with a DragKind of dkDock.
	OnStartDrag (see page 2940)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

53.1.4.3.1 TOutline.OnClick Event

Occurs when the user clicks the control.

Pascal

```
property OnClick;
```

C++

```
__property OnClick;
```

Description

Use the OnClick event handler to respond when the user clicks the control. If the control has an associated action, and that action has an OnExecute method, the action's OnExecute method responds to click events unless it is superseded by an OnClick event handler.

Usually OnClick occurs when the user presses and releases the left mouse button with the mouse pointer over the control. This event can also occur when:

- The user selects an item in a grid, outline, list, or combo box by pressing an arrow key.
- The user presses Spacebar while a button or check box has focus.
- The user presses Enter when the active form has a default button (specified by the Default property).

- The user presses Esc when the active form has a cancel button (specified by the Cancel property).
- The user presses the accelerator key for a button or check box. For example, if the value of the Caption property of a check box is '&Bold', the B is underlined at runtime and the OnClick event of the check box is triggered when the user presses Alt+B. However, focus does not move to the control in these instances.
- The Checked property of a radio button is set to true.
- The value of the Checked property of a check box is changed.
- The Click method of a menu item is called.

For a form, an OnClick event occurs when the user clicks a blank area of the form or on a disabled component.

C++ Examples:

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
    pPage->PageControl = PageControl1;
    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this); // The owner will clean this up.
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
```

```
var
  i: Integer;
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
    begin
      PageControl := PageControl1;
      Caption := 'TabSheet' + IntToStr(i);
      with TEdit.Create(Self) do
      begin
        Parent := PageControl1.Pages[i];
        Left := Random(PageControl1.ActivePage.ClientWidth - Width);
        Top := Random(PageControl1.ActivePage.ClientHeight - Height);
      end;
    end;
    PageControl1Change(Sender);
end;
```

See Also

OnDblClick

Cancel

Default

Click (see page 1375)

OnExecute

Action

Adding Silent Video Clips to an Application

53.1.4.3.2 TOutline.OnCollapse Event

Occurs immediately after a node in the outline is collapsed.

Pascal

```
property OnCollapse: EOutlineChange;
```

C++

```
__property EOutlineChange OnCollapse;
```

Description

Write an OnCollapse event handler to perform special processing when a node in the outline is collapsed. The event handler supplies the index of the node that collapsed.

See Also

OnExpand (see page 2873)

53.1.4.3.3 TOutline.OnContextPopup Event

Occurs when the user right-clicks the control or otherwise invokes the popup menu (such as using the keyboard).

Pascal

```
property OnContextPopup;
```

C++

```
__property OnContextPopup;
```

Description

The OnContextPopup handler is called when the user uses the mouse or keyboard to request a popup menu. The OnContextPopup event is generated by a WM_CONTEXTMENU message, which is itself generated by the user clicking the right mouse button or by pressing SHIFT+F10 or the Applications key.

This event is especially useful when the control does not have an associated popup menu (the PopupMenu property is not set) or if the AutoPopup property of the control's associated popup menu is false. However, the OnContextPopup can also be used to override the automatic context menu that appears when the control has an associated popup menu with an AutoPopup property of true. In this last case, if the event handler displays its own menu, it should set the Handled parameter to true to suppress the default context menu.

The handler's MousePos parameter indicates the position of the mouse, in client coordinates.. If the event was not generated by a mouse click, MousePos is (-1,-1).

Note: Parent controls receive an OnContextPopup event before their child controls. In addition, for many child controls, the default window procedure causes the parent control to receive an OnContextPopup event after the child control. As a result, when parent controls do not set Handled to true in an OnContextPopup event handler, the event handler may be called multiple times for each context menu invocation.

See Also

TPopupMenu (see page 1430)

PopupMenu

AutoPopup (see page 1437)

53.1.4.3.4 TOutline.OnDblClick Event

Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.

Pascal

```
property OnDblClick;
```

C++

```
__property OnDblClick;
```

Description

Use the OnDblClick event to respond to mouse double-clicks.

See Also

OnClick

Responding to User Actions at Runtime

53.1.4.3.5 TOutline.OnDragDrop Event

Occurs when the user drops an object being dragged.

Pascal

```
property OnDragDrop;
```

C++

```
__property OnDragDrop;
```

Description

Use the OnDragDrop event handler to specify what happens when the user drops an object. The Source parameter of the OnDragDrop event is the object being dropped, and the Sender is the control the object is being dropped on. The X and Y parameters are the coordinates of the mouse positioned over the control.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}

void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
```

```
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
}
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  TForm1(Source).Left := X;
  TForm1(Source).Top := Y;
end;

procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := False;
  if (Source is TLabel) and (State = dsDragEnter) then
    (Source as TLabel).EndDrag(False);
end;
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
```

```
  dropped label.
  }

  procedure TForm1.FormCreate(Sender: TObject);
  begin
    ListBox1.Items.Add('Not');
    ListBox1.Items.Add('In');
    ListBox1.Items.Add('Alphabetical');
    ListBox1.Items.Add('Order');
  end;

  // This OnDragOver event handler permits the list box to
  // accept a dropped label:

  procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
    State: TDragState; var Accept: Boolean);
  begin
    Accept := Source is TLabel;
  end;

  // This OnDragDrop event handler implements the drop behavior.

  procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
  begin
    if (Sender is TListBox) and (Source is TLabel) then
    begin
      with Sender as TListBox do
      begin
        Font := (Source as TLabel).Font;
        Color := (Source as TLabel).Color;
        DoubleBuffered := true;
        Color := clWindow;
      end;
    end;
  end;
```

See Also

BeginDrag

DragDrop

Dragging

EndDrag

OnDragOver

OnEndDrag

OnStartDrag

TDragObject

53.1.4.3.6 TOutline.OnDragOver Event

Occurs when the user drags an object over a control.

Pascal

```
  property OnDragOver;
```

C++

```
  __property OnDragOver;
```

Description

Use an OnDragOver event to signal that the control can accept a dragged object so the user can drop or dock it.

Within the OnDragOver event handler, change the Accept parameter to false to reject the dragged object. Leave Accept as true to allow the user to drop or dock the dragged object on the control.

To change the shape of the cursor, indicating that the control can accept the dragged object, change the value of the DragCursor property for the control before the OnDragOver event occurs.

The Source is the object being dragged, the Sender is the potential drop or dock site, and X and Y are screen coordinates in pixels. The State parameter specifies how the dragged object is moving over the control.

Note: Within the OnDragOver event handler, the Accept parameter defaults to true. However, if an OnDragOver event handler is not supplied, the control rejects the dragged object, as if the Accept parameter were changed to false.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}

void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
```

```
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
}
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  TForm1(Source).Left := X;
  TForm1(Source).Top := Y;
end;

procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
```

```
    State: TDragState; var Accept: Boolean);
  begin
    Accept := False;
    if (Source is TLabel) and (State = dsDragEnter) then
      (Source as TLabel).EndDrag(False);
  end;
  {
  This code requires a list box and three labels, each with a
  different font and color.  The DragMode property for each of
  the labels is dmAutomatic.  The user can select a label and
  drag it to a list box and drop it. When the label is dropped,
  the items in the list box assume the color and font of the
  dropped label.
  }

  procedure TForm1.FormCreate(Sender: TObject);
  begin
    ListBox1.Items.Add('Not');
    ListBox1.Items.Add('In');
    ListBox1.Items.Add('Alphabetical');
    ListBox1.Items.Add('Order');
  end;

  // This OnDragOver event handler permits the list box to
  // accept a dropped label:

  procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
    State: TDragState; var Accept: Boolean);
  begin
    Accept := Source is TLabel;
  end;

  // This OnDragDrop event handler implements the drop behavior.

  procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
  begin
    if (Sender is TListBox) and (Source is TLabel) then
    begin
      with Sender as TListBox do
      begin
        Font := (Source as TLabel).Font;
        Color := (Source as TLabel).Color;
        DoubleBuffered := true;
        Color := clWindow;
      end;
    end;
  end;
```

See Also

DragCursor

OnDragDrop

OnEndDrag

OnStartDrag

TDragObject

DragKind

OnEndDock

OnDockDrop

OnStartDock

53.1.4.3.7 TOutline.OnDrawItem Event

Occurs when a node must be drawn in an owner-draw outline.

Pascal

```
property OnDrawItem: TDrawItemEvent;
```

C++

```
__property TDrawItemEvent OnDrawItem;
```

Description

When the Style property is set to otOwnerDraw, specify an OnDrawItem event handler to render the image of each node in the outline. An OnDrawItem event handler can display items other than the Text of a node and the standard bitmaps specified in the PictureClosed, PictureOpen, PictureMinus, PicturePlus, and PictureLeaf properties.

Draw the image of the node specified by the Index parameter onto the canvas of the outline.

See Also

Canvas

PictureClosed (see page 2868)

PictureLeaf (see page 2868)

PictureMinus (see page 2869)

PictureOpen (see page 2869)

PicturePlus (see page 2870)

Style (see page 2872)

Text (see page 2955)

53.1.4.3.8 TOutline.OnEndDock Event

Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.

Pascal

```
property OnEndDock;
```

C++

```
__property OnEndDock;
```

Description

Use OnEndDock to specify actions or special processing that when a drag-and-dock operation stops.

See Also

DragKind

OnDragOver

OnDockDrop

OnStartDock

53.1.4.3.9 TOutline.OnEndDrag Event

Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.

Pascal

```
property OnEndDrag;
```

C++

```
__property OnEndDrag;
```

Description

Use the OnEndDrag event handler to specify any special processing that occurs when dragging stops.

C++ Examples:

```
/*
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
*/
void __fastcall TForm1::Label1EndDrag(TObject *Sender, TObject *Target, int X,
      int Y)
{
  if (Sender->ClassNameIs("TLabel"))
  {
    AnsiString S = (dynamic_cast<TLabel *>(Sender))->Name + " was dropped and ";
    if (Target)
      S = S + "accepted!";
    else
      S = S + "rejected.";
    Status->Caption = S;
  }
}
```

Delphi Examples:

```
{
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
}
procedure TForm1.Label1EndDrag(Sender, Target: TObject; X, Y: Integer);
var
  S: string;
begin
  S := (Sender as TLabel).Name + ' was dropped... and ';
  if Target <> nil then
    S := S + 'accepted!'
  else
    S := S + 'rejected!';
  Status.Caption := S;
end;
```

See Also

OnDragDrop

OnDragOver

OnStartDrag

DragKind

53.1.4.3.10 TOutline.OnEnter Event

Occurs when a control receives the input focus.

Pascal

```
property OnEnter;
```

C++

```
__property OnEnter;
```

Description

Use the OnEnter event handler to cause any special processing to occur when a control becomes active.

The OnEnter event does not occur when switching between forms or between another application and the application that includes the control.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnEnter event occurs for the container before the OnEnter event of the contained control.

Similarly, an OnExit event of the container occurs after the OnExit event of the control in a container when focus moves to another control outside the container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

C++ Examples:

```
/*
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
*/
TWinControl *myActiveControl;

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  FontDialog1->Options << fdApplyButton;
  FontDialog1->Execute();
}

void __fastcall TForm1::FontDialog1Apply(TObject *Sender, HWND Wnd)
{
  if (myActiveControl->ClassNameIs("TEdit"))
   (dynamic_cast<TEdit *>(myActiveControl))->Font->Assign(FontDialog1->Font);
  else if (myActiveControl->ClassNameIs("TRichEdit"))
    (dynamic_cast<TRichEdit *>(myActiveControl))->SelAttributes->Assign(FontDialog1->Font);
  else
    MessageBeep(0);
}

void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  myActiveControl = ActiveControl;
```

```
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
   FontDialog1.Options := FontDialog1.Options + [fdApplyButton];
   FontDialog1.Execute;
end;

procedure TForm1.Edit1Enter(Sender: TObject);
begin
  myActiveControl := ActiveControl;
end;

procedure TForm1.FontDialog1Apply(Sender: TObject; Wnd: HWND);
begin
  if myActiveControl is TEdit then
    with myActiveControl as TEdit do
       Font.Assign(TFontDialog(Sender).Font)
  else if myActiveControl is TRichEdit then
    with myActiveControl as TRichEdit do
      SelAttributes.Assign(TFontDialog(Sender).Font)
  else
    Beep;
end;
{
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
```

```
  inactive, the color of the control returns to the Windows
  system color for a window.  These event handlers could also
  be shared.
  }
  procedure TForm1.Edit1Enter(Sender: TObject);
  begin
    Edit1.Color := clYellow;
  end;

  procedure TForm1.Edit1Exit(Sender: TObject);
  begin
    Edit1.Color := clWindow;
  end;

  procedure TForm1.Memo1Enter(Sender: TObject);
  begin
    Memo1.Color := clYellow;
  end;

  procedure TForm1.Memo1Exit(Sender: TObject);
  begin
    Memo1.Color := clWindow;
  end;
```

See Also

OnExit

ActiveControl

OnActivate

Using a Single Navigator for Multiple Datasets

53.1.4.3.11 TOutline.OnExit Event

Occurs when the input focus shifts away from one control to another.

Pascal

```
property OnExit;
```

C++

```
__property OnExit;
```

Description

Use the OnExit event handler to provide special processing when the control ceases to be active.

The OnExit event does not occur when switching between forms or between another application and your application.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnExit event occurs for the control inside the container before the OnExit event of the container.

Similarly, an OnEnter event of the container occurs before the OnEnter event of the control in a container when focus moves to a control inside a container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

Note: In some control classes the ActiveControl property updates before the OnExit event occurs.

C++ Examples:

```
/*
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255 ,
as this validation test cannot be accomplished
using the edit mask property alone.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  // initialize the maskedit boxes to contain the empty string
  MaskEdit1->Text = "";
  MaskEdit2->Text = "";
  MaskEdit3->Text = "";

  // use the Courier font in the maskedit boxes
  MaskEdit1->Font->Name = "Courier";
  MaskEdit2->Font->Name = "Courier";
  MaskEdit3->Font->Name = "Courier";

  // initialize the mask of each maskedit box
  // according to IP address coding conventions
  MaskEdit1->EditMask = "!099.099.099.099;1; ";
  MaskEdit2->EditMask = "!099.099.099.099;1; ";
  MaskEdit3->EditMask = "!099.099.099.099;1; ";

  MaskEdit2->OnExit = MaskEdit2Exit;
}

void __fastcall TForm1::MaskEdit2Exit(TObject *Sender)
{
  // extract the net and host address from the IP
  String IP = MaskEdit1->Text;
  int net1 = IP.SubString(0, 3).TrimRight().ToInt();
  int net2 = IP.SubString(5, 3).TrimRight().ToInt();
  int host1 = IP.SubString(9, 3).TrimRight().ToInt();
  int host2 = IP.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (net1 < 0 || net1 > 255 ||
      net2 < 0 || net2 > 255 ||
      host1 < 0 || host1 > 255 ||
      host2 < 0 || host2 > 255)
    throw(Exception("Not a valid IP address."));

  // extract the net and host mask from the subnet mask
  String mask = MaskEdit2->Text;
  int netmask1 = mask.SubString(0, 3).TrimRight().ToInt();
  int netmask2 = mask.SubString(5, 3).TrimRight().ToInt();
  int hostmask1 = mask.SubString(9, 3).TrimRight().ToInt();
  int hostmask2 = mask.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (netmask1 < 0 || netmask1 > 255 ||
      netmask2 < 0 || netmask2 > 255 ||
```

```
        hostmask1 < 0 || hostmask1 > 255 ||
        hostmask2 < 0 || hostmask2 > 255)
      throw(Exception("Not a valid subnet mask."));

    // compute the subnet address using bit-wise AND
    int sub_net1 = net1 & netmask1;
    int sub_net2 = net2 & netmask2;
    int sub_host1 = host1 & hostmask1;
    int sub_host2 = host2 & hostmask2;

    // display the subnet address
    MaskEdit3->Text =
      String(sub_net1) + "." + String(sub_net2) + "." +
      String(sub_host1) + "." + String(sub_host2);
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255,
as this validation test cannot be accomplished
using the edit mask property alone.
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  { initialize the maskedit boxes to contain the empty string }
  MaskEdit1.Text := '';
  MaskEdit2.Text := '';
  MaskEdit3.Text := '';
```

```
    { use the Courier font in the maskedit boxes }
    MaskEdit1.Font.Name := 'Courier';
    MaskEdit2.Font.Name := 'Courier';
    MaskEdit3.Font.Name := 'Courier';

    {
    initialize the mask of each maskedit box
    according to IP address coding conventions
    }
    MaskEdit1.EditMask := '!099.099.099.099;1; ';
    MaskEdit2.EditMask := '!099.099.099.099;1; ';
    MaskEdit3.EditMask := '!099.099.099.099;1; ';

    MaskEdit2.OnExit := MaskEdit2Exit;
  end;

  procedure TForm1.MaskEdit2Exit(Sender: TObject);
  var
    net1, net2, host1, host2,
    netmask1, netmask2, hostmask1, hostmask2,
    sub_net1, sub_net2, sub_host1, sub_host2: Integer;

    IP, mask: String;

  begin
    // extract the net and host address from the IP
    IP := MaskEdit1.Text;
    net1 := StrToInt(TrimRight(Copy(IP, 0, 3)));
    net2 := StrToInt(TrimRight(Copy(IP, 5, 3)));
    host1 := StrToInt(TrimRight(Copy(IP, 9, 3)));
    host2 := StrToInt(TrimRight(Copy(IP, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((net1 < 0) Or (net1 > 255) Or
        (net2 < 0) Or (net2 > 255) Or
        (host1 < 0) Or (host1 > 255) Or
        (host2 < 0) Or (host2 > 255)) then
      raise EArgumentException.Create('Not a valid IP address.');

    // extract the net and host mask from the subnet mask
    mask := MaskEdit2.Text;
    netmask1 := StrToInt(TrimRight(Copy(mask, 0, 3)));
    netmask2 := StrToInt(TrimRight(Copy(mask, 5, 3)));
    hostmask1 := StrToInt(TrimRight(Copy(mask, 9, 3)));
    hostmask2 := StrToInt(TrimRight(Copy(mask, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((netmask1 < 0) Or (netmask1 > 255) Or
        (netmask2 < 0) Or (netmask2 > 255) Or
        (hostmask1 < 0) Or (hostmask1 > 255) Or
        (hostmask2 < 0) Or (hostmask2 > 255)) then
      raise EArgumentException.Create('Not a valid subnet mask.');

    // compute the subnet address using bit-wise AND
    sub_net1 := net1 And netmask1;
    sub_net2 := net2 And netmask2;
    sub_host1 := host1 And hostmask1;
    sub_host2 := host2 And hostmask2;

    // display the subnet address
    MaskEdit3.Text :=
      IntToStr(sub_net1) + '.' + IntToStr(sub_net2) + '.' +
      IntToStr(sub_host1) + '.' + IntToStr(sub_host2);
  end;
  {
  This example uses an edit box and a memo control on a form.
```

```
  When either the edit box or the memo is the active control,
  it is colored yellow. When the active control becomes
  inactive, the color of the control returns to the Windows
  system color for a window.  These event handlers could also
  be shared.
  }
  procedure TForm1.Edit1Enter(Sender: TObject);
  begin
    Edit1.Color := clYellow;
  end;

  procedure TForm1.Edit1Exit(Sender: TObject);
  begin
    Edit1.Color := clWindow;
  end;

  procedure TForm1.Memo1Enter(Sender: TObject);
  begin
    Memo1.Color := clYellow;
  end;

  procedure TForm1.Memo1Exit(Sender: TObject);
  begin
    Memo1.Color := clWindow;
  end;
```

See Also

OnEnter

ActiveControl

53.1.4.3.12 TOutline.OnExpand Event

Occurs immediately after a node in the outline is expanded.

Pascal

```
property OnExpand: EOutlineChange;
```

C++

```
__property EOutlineChange OnExpand;
```

Description

Write an OnExpand event handler to perform special processing when a node in the outline is expanded. The event handler identifies the node that was just expanded.

See Also

OnCollapse (see page 2873)

53.1.4.3.13 TOutline.OnKeyDown Event

Occurs when a user presses any key while the control has focus.

Pascal

```
property OnKeyDown;
```

C++

```
__property OnKeyDown;
```

Description

Use the OnKeyDown event handler to specify special processing to occur when a key is pressed. The OnKeyDown handler can respond to keyboard keys, including function keys and keys combined with the `Shift`, `Alt`, and `Ctrl` keys, and pressed mouse buttons.

Note: TCheckBox does not fire OnKeyDown events for arrow keys.

The TKeyEvent type points to a method that handles keyboard events.

The Key parameter is the key on the keyboard. For non-alphanumeric keys, use virtual key codes to determine the key pressed. For more information, see Virtual Key codes.

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

An application gets Windows WM_KEYDOWN messages for all keys when the user presses a key. These messages indirectly fire the OnKeyDown event. If you set the Key to 0 there you prevent only any further processing of this message. But for keys that generate characters Windows also produces WM_CHAR. At the time your OnKeyDown event fires, the WM_CHAR message for the key will already be in the message queue. Setting Key to 0 does not stop it from being delivered, so it fires the OnKeyPress event. You must set the Key to #0 to prevent the control from intercepting the WM_CHAR message.

This method of organizing key processing has advantages. Code that only deals with characters, including control characters like #13 for carriage return, #3 for Ctrl-C and so on, should go into the OnKeyPress event. Code that deals with keys that do not generate characters should be put into the OnKeyDown event.

C++ Examples:

```
/*
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
*/
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  if (Key == VK_F1)
  {
    FormColor = Form1->Color;
    Form1->Color = clAqua;
    if (ComboBox1->Text == "KeyDown")
      ComboBox1->Text = "KeyDown2";
    else
      ComboBox1->Text = "KeyDown";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}

void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key, TShiftState Shift)
{
  if (Key == VK_F1)
  {
    Form1->Color = FormColor;
    if (ComboBox1->Text == "KeyUp")
      ComboBox1->Text = "KeyUp2";
    else
      ComboBox1->Text = "KeyUp";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
```

```
}
/*
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
*/
#include <Printers.hpp>

void __fastcall TForm1::FormCreate(TObject *Sender)
{
   KeyPreview = True;

}

void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)

{
  if (Key == VK_ESCAPE && Printer()->Printing)
  {
    Printer()->Abort();
    MessageDlg("Printing aborted", mtInformation,
               TMsgDlgButtons() << mbOK,0);
  }
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMemo *Memo1 = new TMemo(Form1); // The parent (Form1) will clean up this object.
  Memo1->Parent = Form1;
  Memo1->Visible = True;
  Memo1->Width = 700;
  if (OpenDialog1->Execute())
  {
    Memo1->Lines->LoadFromFile(OpenDialog1->FileName);

    Printer()->BeginDoc();
    int X = 200;
    int Y = 200;
    for (int I = 0; I <= 10; I++)
      if (!Printer()->Aborted)
      {
        Printer()->Canvas->TextOut(X, Y, Memo1->Lines->Strings[I]);
        Y = Y + 80;
        if (Y > (Printer()->PageHeight - 300))
        {
          Y = 200;
          Printer()->NewPage();
          Sleep(1000);  // to give you time to abort!
        }
      }
    if (!Printer()->Aborted) Printer()->EndDoc();
  }
}
```

Delphi Examples:

```
{
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
}
var
  FormColor: TColor;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  KeyPreview := True;
  FormColor := Form1.Color;
end;

procedure TForm1.FormKeyDown(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := clAqua;
    if (ComboBox1.Text = 'KeyDown') then
      ComboBox1.Text := 'KeyDown2'
    else
      ComboBox1.Text := 'KeyDown';
    WriteLn(ComboBox1.Text);
  end;
end;

procedure TForm1.FormKeyUp(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := FormColor;
    if (ComboBox1.Text = 'KeyUp') then
      ComboBox1.Text := 'KeyUp2'
    else
      ComboBox1.Text := 'KeyUp';
    WriteLn(ComboBox1.Text);
  end;
end;
{
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
}

uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  I, X, Y: Integer;
  Memo1 : TMemo;
  r: TRect;
begin
  Memo1 := TMemo.Create(Form1);
  Memo1.Parent := Form1;
  Memo1.Visible := True;
  Memo1.Width := 700;
  if (OpenDialog1.Execute) then
  begin
    Memo1.Lines.LoadFromFile(OpenDialog1.FileName);

    Printer.BeginDoc;
    X := 200;
    Y := 200;
    for I := 0 to 140 do
      if (not Printer.Aborted) then
      begin
        Printer.Canvas.TextOut(X, Y, Memo1.Lines[I]);
        Y := Y + 80;
        if (Y > (Printer.PageHeight - 300)) then
        begin
          Y := 200;
```

```
            Printer.NewPage;
            Sleep(1000);  // to give you time to abort!
          end;
        end;
      if (not Printer.Aborted) then Printer.EndDoc;
    end;
  end;

  procedure TForm1.FormCreate(Sender: TObject);
  begin
     KeyPreview := True;
  end;

  procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
  begin
  if (Key=VK_ESCAPE) and Printer.Printing then
    begin
    Printer.Abort;
    MessageDlg('Printing aborted', mtInformation, [mbOK],0);
    end;
  end;
```

See Also

KeyPreview

OnKeyPress

OnKeyUp

53.1.4.3.14 TOutline.OnKeyPress Event

Occurs when a key is pressed.

Pascal

```
property OnKeyPress;
```

C++

```
__property OnKeyPress;
```

Description

Use the OnKeyPress event handler to make something happen as a result of a single character key press.

The Key parameter in the OnKeyPress event handler is of type Char; therefore, the OnKeyPress event registers the ASCII character of the key pressed. Keys that don't correspond to an ASCII Char value (Shift or F1, for example) don't generate an OnKeyPress event. Key combinations (such as Shift+A), generate only one OnKeyPress event (for this example, Shift+A results in a Key value of "A" if Caps Lock is off). To respond to non-ASCII keys or key combinations, use the OnKeyDown or OnKeyUp event handlers.

An application gets Windows WM_KEYDOWN messages for all keys when the user presses a key. These messages indirectly fire the OnKeyDown event. If you set the Key to 0 there you prevent only any further processing of this message. But for keys that generate characters Windows also produces WM_CHAR. At the time your OnKeyDown event fires, the WM_CHAR message for the key will already be in the message queue. Setting Key to 0 does not stop it from being delivered, so it fires the OnKeyPress event. You must set the Key to #0 to prevent the control from intercepting the WM_CHAR message.

This method of organizing key processing has advantages. Code that only deals with characters, including control characters like #13 for carriage return, #3 for Ctrl-C and so on, should go into the OnKeyPress event. Code that deals with keys that do not generate characters should be put into the OnKeyDown event.

C++ Examples:

```
/*
This event handler displays a message dialog box specifying
which key was pressed:
*/

void __fastcall TForm1::FormKeyPress(TObject *Sender, char &Key)
{
  char keyString[25];
  keyString[0] = Key;
  strcpy(&keyString[1], " Was Pressed");
  TMsgDlgButtons() << mbOK;
  MessageDlg(AnsiString(keyString), TMsgDlgType(mtInformation), TMsgDlgButtons() << mbOK, 0);
}
```

Delphi Examples:

```
{
This event handler displays a message dialog box specifying
which key was pressed:
}
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  MessageDlg(Key + ' has been pressed', mtInformation, [mbOK], 0)
end;
```

See Also

OnKeyDown

OnKeyUp

KeyPreview

53.1.4.3.15 TOutline.OnKeyUp Event

Occurs when the user releases a key that has been pressed.

Pascal

```
property OnKeyUp;
```

C++

```
__property OnKeyUp;
```

Description

Use the OnKeyUp event handler to provide special processing that occurs when a key is released. The OnKeyUp handler can respond to all keyboard keys, including function keys and keys combined with the Shift, Alt, and Ctrl keys.

The TKeyEvent type points to a method that handles keyboard events. The Key parameter is the key on the keyboard. For non-alphanumeric keys, you must use virtual key codes to determine the key pressed. For more information, see Virtual Key codes.

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

C++ Examples:

```
/*
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
*/
```

```
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  if (Key == VK_F1)
  {
    FormColor = Form1->Color;
    Form1->Color = clAqua;
    if (ComboBox1->Text == "KeyDown")
      ComboBox1->Text = "KeyDown2";
    else
      ComboBox1->Text = "KeyDown";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}

void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key, TShiftState Shift)
{
  if (Key == VK_F1)
  {
    Form1->Color = FormColor;
    if (ComboBox1->Text == "KeyUp")
      ComboBox1->Text = "KeyUp2";
    else
      ComboBox1->Text = "KeyUp";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}
```

Delphi Examples:

```
{
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
}
var
  FormColor: TColor;
procedure TForm1.FormCreate(Sender: TObject);
begin
  KeyPreview := True;
  FormColor := Form1.Color;
end;

procedure TForm1.FormKeyDown(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := clAqua;
    if (ComboBox1.Text = 'KeyDown') then
      ComboBox1.Text := 'KeyDown2'
    else
      ComboBox1.Text := 'KeyDown';
    WriteLn(ComboBox1.Text);
  end;
end;

procedure TForm1.FormKeyUp(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
```

```
  begin
    Form1.Color := FormColor;
    if (ComboBox1.Text = 'KeyUp') then
      ComboBox1.Text := 'KeyUp2'
    else
      ComboBox1.Text := 'KeyUp';
    WriteLn(ComboBox1.Text);
  end;
end;
```

See Also

OnKeyDown

OnKeyPress

53.1.4.3.16 TOutline.OnMouseActivate Event

Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.

Pascal

```
property OnMouseActivate;
```

C++

```
__property OnMouseActivate;
```

Description

Use the OnMouseActivate event handler to implement any special processing that should occur as a result of pressing a mouse button on a control when the parent top-level form is not active.

The event type is:

```
TMouseActivateEvent = procedure (Sender: TObject; Button: TMouseButton; ShiftState:
TShiftState;
X, Y: Integer; HitTest: Integer; var MouseActivate: TMouseActivate) of object;
```

When you click on a control and the parent top-level form is not active, this event fires on the control that the mouse cursor is over. MouseActivate is a protected virtual function in TControl. MouseActivate can be overridden in custom control descendants to provide special built-in processing of the OnMouseActivate events. If you leave the default value of MouseActivate as maDefault, the parent control fires the OnMouseActivate event and repeats the process all the way up to the top-level form. If no control in the chain sets the value of MouseActivate, the behavior is the same as if a control set TMouseActivate to maActivate. At any point, a control can set the value of TMouseActivate and the parent processing stops.

Some controls set focus to themselves when they get a button down message so, in some cases, setting maNoActivate appears to have no effect. For instance, a TButton control sets focus to itself in the WM_LBUTTONDOWN message regardless of whether or not TMouseActivate is set to maNoActivate. In this case, setting maNoActivateAndEat will work because the top-level form is not activated and the WM_LBUTTONDOWN message is suppressed.

The HitTest parameter is the hit test value obtained from the WM_NCHITTEST message. See Windows.pas for valid values and their meanings. In most cases this value will be HTCLIENT, which means that the user clicked in the client area of a control. However, in the case of a top-level form, this can take other values such as HTCAPTION, or HTBOTTOM. This allows the code to decide which value to set for MouseActivate, based on the HitTest code. For example, the following code in the OnMouseActivate event handler for a TForm forces the user to click into a non-client area to activate the form:

```
if HitTest = HTCLIENT then
    MouseActivate := maNoActivateAndEat;
```

Even though this event is tied to how Windows processes WM_MOUSEACTIVATE, the actual handling of the message is done at the TControl level, which is the point in the VCL hierarchy where the control has no window handle. The OnMouseActivate event is then simulated since the all TControl descendants also have OnMouseDown, OnMouseMove, and OnMouseUp events.

This information is applicable to both VCL Win32 and VCL.NET.

See Also

OnClick

OnMouseMove

OnMouseUp

MouseCapture

53.1.4.3.17 TOutline.OnMouseDown Event

Occurs when the user presses a mouse button with the mouse pointer over a control.

Pascal

```
property OnMouseDown;
```

C++

```
__property OnMouseDown;
```

Description

Use the OnMouseDown event handler to implement any special processing that should occur as a result of pressing a mouse button.

The OnMouseDown event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. X and Y are the pixel coordinates of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
```

```
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  Form1.Canvas.Rectangle(StartX, StartY, X, Y);
  StatusBar1.Panels[0].Text := '';
  StatusBar1.Panels[1].Text := '';
  StatusBar1.Panels[2].Text := '';
  StatusBar1.Panels[3].Text := '';
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if ssLeft in Shift then { make sure button is down }
  begin
    if Y > StartY then
```

```
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
        end
      else
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
        end;
      if X > StartX then
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
        end
      else
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
        end;
    end;
  end;
```

See Also

OnClick

OnMouseMove

OnMouseUp

MouseCapture

53.1.4.3.18 TOutline.OnMouseEnter Event

Occurs when the user moves the mouse into a control.

Pascal

```
property OnMouseEnter;
```

C++

```
__property OnMouseEnter;
```

Description

Use the OnMouseEnter event handler to implement any special processing that should occur as a result of moving the mouse into a control when the parent top-level form is not active.

53.1.4.3.19 TOutline.OnMouseLeave Event

Occurs when the user moves the mouse outside of a control.

Pascal

```
property OnMouseLeave;
```

C++

```
__property OnMouseLeave;
```

Description

Use the OnMouseLeave event handler to implement any special processing that should occur as a result of moving the mouse outside of a control when the parent top-level form is not active.

53.1.4.3.20 TOutline.OnMouseMove Event

Occurs when the user moves the mouse pointer while the mouse pointer is over a control.

Pascal

```
property OnMouseMove;
```

C++

```
__property OnMouseMove;
```

Description

Use the OnMouseMove event handler to respond when the mouse pointer moves after the control has captured the mouse.

Use the Shift parameter of the OnMouseDown event handler, to determine to the state of the shift keys and mouse buttons. Shift keys are the Shift, Ctrl, and Alt keys or shift key-mouse button combinations. X and Y are pixel coordinates of the new location of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
```

```
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  Form1.Canvas.Rectangle(StartX, StartY, X, Y);
  StatusBar1.Panels[0].Text := '';
  StatusBar1.Panels[1].Text := '';
  StatusBar1.Panels[2].Text := '';
  StatusBar1.Panels[3].Text := '';
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if ssLeft in Shift then { make sure button is down }
  begin
    if Y > StartY then
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
      end
    else
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
      end;
    if X > StartX then
      begin
      StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
      StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
```

```
      end
    else
      begin
      StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
      StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
      end;
  end;
end;
```

See Also

OnMouseDown

OnMouseUp

MouseCapture

53.1.4.3.21 TOutline.OnMouseUp Event

Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.

Pascal

```
property OnMouseUp;
```

C++

```
__property OnMouseUp;
```

Description

Use an OnMouseUp event handler to implement special processing when the user releases a mouse button.

The OnMouseUp event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. X and Y are the pixel coordinates of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
```

```
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  Form1.Canvas.Rectangle(StartX, StartY, X, Y);
  StatusBar1.Panels[0].Text := '';
  StatusBar1.Panels[1].Text := '';
  StatusBar1.Panels[2].Text := '';
  StatusBar1.Panels[3].Text := '';
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
```

```
    if ssLeft in Shift then { make sure button is down }
    begin
      if Y > StartY then
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
        end
      else
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
        end;
      if X > StartX then
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
        end
      else
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
        end;
    end;
  end;
```

See Also

OnMouseDown

OnMouseMove

Responding to a Mouse-up Action

53.1.4.3.22 TOutline.OnStartDock Event

Occurs when the user begins to drag a control with a DragKind of dkDock.

Pascal

```
property OnStartDock;
```

C++

```
__property OnStartDock;
```

Description

Use the OnStartDock event handler to implement special processing when the user starts a drag-and-dock operation by dragging the control.

The OnStartDock event handler can create a TDragDockObjectEx object for the DragObject parameter to specify the appearance of the dragging rectangle and how the dragged control interacts with potential docking sites. If you return TDragDockObjectEx as the drag object, there is no need to call the Free method for the DragObject when dragging is over. If you use TDragDockObject, your application is responsible for freeing the drag object.

If the OnStartDock event handler sets the DragObject parameter to nil (Delphi) or NULL (C++), a TDragDockObject object is automatically created.

Note: This information applies to Win32 applications only.

See Also

OnEndDock

TDragDockObjectEx

DragKind

OnDragOver

OnDockDrop

53.1.4.3.23 TOutline.OnStartDrag Event

Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

Pascal

```
property OnStartDrag;
```

C++

```
__property OnStartDrag;
```

Description

Use the OnStartDrag event handler to implement special processing when the user starts to drag the control or an object it contains. OnStartDrag only occurs if DragKind is dkDrag.

Sender is the control that is about to be dragged, or that contains the object about to be dragged.

The OnStartDrag event handler can create a TDragControlObjectEx instance for the DragObject parameter to specify the drag cursor, or, optionally, a drag image list. If you create a TDragControlObjectEx instance, there is no need to call the Free method for the DragObject when dragging is over. If you create, instead, a TDragControlObject instance, your application is responsible for freeing the drag object instance.

If the OnStartDrag event handler sets the DragObject parameter to nil (Delphi) or NULL (C++), a TDragControlObject object is automatically created and dragging begins on the control itself.

Note: On some controls, such as TRichEdit, the underlying Windows control handles internal drag operations. For these controls, there are no OnStartDrag or OnEndDrag events for drag operations within the control.

See Also

OnEndDrag

TDragControlObjectEx

DragKind

OnDragOver

OnDragDrop

53.1.5 TOutlineNode Class

TOutlineNode represents a node in a TCustomOutline (see page 2838) object.

Class Hierarchy

Pascal

```
TOutlineNode = class(TPersistent);
```

C++

```
class TOutlineNode : public TPersistent;
```

File

Outline

Description

Use TOutlineNode to describe a node in a Win 3.1 outline. Each outline node appears as a line, or row, of the outline. An outline node can have from 0 to 16368 subnodes, which are subordinate to the parent node in the outline structure.

See Also

TOutline (see page 2874)

53.1.5.1 TOutlineNode Members

The following tables list the members exposed by TOutlineNode.

Protected Methods

	Name	Description
	AddNode (see page 2944)	This is AddNode, a member of class TOutlineNode.
	GetDataItem (see page 2946)	This is GetDataItem, a member of class TOutlineNode.
	GetNodeAtIndex (see page 2947)	This is GetNodeAtIndex, a member of class TOutlineNode.
	GetRowOfNode (see page 2948)	This is GetRowOfNode, a member of class TOutlineNode.
	GetTextItem (see page 2948)	This is GetTextItem, a member of class TOutlineNode.
	GetVisibleNode (see page 2949)	This is GetVisibleNode, a member of class TOutlineNode.
	HasAsParent (see page 2949)	This is HasAsParent, a member of class TOutlineNode.
	InsertNode (see page 2949)	This is InsertNode, a member of class TOutlineNode.
	InternalRemove (see page 2949)	This is InternalRemove, a member of class TOutlineNode.
	Remove (see page 2950)	This is Remove, a member of class TOutlineNode.
	WriteNode (see page 2950)	This is the overview for the WriteNode method overload.

Protected Properties

	Name	Description
R	ExpandCount (see page 2952)	This is ExpandCount, a member of class TOutlineNode.
R	Items (see page 2954)	This is Items, a member of class TOutlineNode.
R	List (see page 2955)	This is List, a member of class TOutlineNode.
R	Outline (see page 2955)	This is Outline, a member of class TOutlineNode.

Public Methods

	Name	Description
	ChangeLevelBy (see page 2944)	Changes the level of an outline node.
	Collapse (see page 2944)	Collapses an outline item.
	Create (see page 2942)	Creates a new TOutlineNode (see page 2940) object.
V	Destroy (see page 2943)	Frees the memory associated with the TOutlineNode (see page 2940) object.
	Expand (see page 2945)	Expands an outline item.
	FullExpand (see page 2945)	Expands an outline node and all its descendants.
	GetDisplayWidth (see page 2946)	Returns the distance, in pixels, from the left edge of the outline to the right edge of the outline node.
	getFirstChild (see page 2946)	Returns the Index (see page 2953) value of the first subnode.
	GetLastChild (see page 2947)	Returns the Index (see page 2953) value of the last subnode.
	GetNextChild (see page 2947)	Returns the Index (see page 2953) value of the next child of an outline node.
	GetPrevChild (see page 2948)	Returns the Index (see page 2953) value of the previous child of an outline node.

	MoveTo (see page 2949)	Moves an outline node from one location within the outline to another.

Public Properties

	Name	Description
	Data (see page 2951)	Points to application-specific data associated with the node.
	Expanded (see page 2952)	Determines whether descendants of the outline node are displayed.
R	FullPath (see page 2952)	Describes an outline node, using its text and the text of all ancestor nodes in the outline.
R	HasItems (see page 2953)	Indicates whether the outline node has any subnodes.
R	Index (see page 2953)	Identifies the node in the Items (see page 2954) array maintained by the outline.
R	IsVisible (see page 2954)	Indicates whether the outline node is visible in the outline.
	Level (see page 2954)	Indicates the level of indentation of a node within the outline.
R	Parent (see page 2955)	Identifies the immediate ancestor of the node.
	Text (see page 2955)	Specifies the string that identifies the outline node.
R	TopItem (see page 2956)	Indicates which outline node is the topmost ancestor of a node.

53.1.5.2 TOutlineNode.Create Constructor

Creates a new TOutlineNode (see page 2940) object.

Pascal

```
constructor Create(AOwner: TCustomOutline);
```

C++

```
__fastcall TOutlineNode(TCustomOutline AOwner);
```

Description

Do not call the constructor for TOutlineNode (see page 2940) directly. Instead, use one of the methods of the outline component in which the node should appear. These methods are as follows:

Method	Action
Add	Creates the node as the last sibling of a specified node.
AddObject	Same as Add, but also initializing the node's Data (see page 2951) property.
AddChild	Creates the node as the last child of a specified node.
AddChildObject	Same as AddChild, but also initializing the node's Data (see page 2951) property.
Insert	Creates the node in the current position of an existing node.
InsertObject	Same as Insert, but also initializing the node's Data (see page 2951) property.

See Also

Data (see page 2951)

Add (see page 2842)

AddObject (see page 2844)

AddChild (see page 2843)

Insert (see page 2853)

AddChildObject (see page 2844)

InsertObject (see page 2853)

53.1.5.3 TOutlineNode.Destroy Destructor

Frees the memory associated with the TOutlineNode (see page 2940) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TOutlineNode();
```

Description

Do not call Destroy directly. The preferred method of deleting nodes is to use the Delete method of the containing outline component.

See Also

System::TObject::Free

Delete (see page 2848)

53.1.5.4 TOutlineNode Methods

The methods of the TOutlineNode class are listed here.

Protected Methods

	Name	Description
	AddNode (see page 2944)	This is AddNode, a member of class TOutlineNode.
	GetDataItem (see page 2946)	This is GetDataItem, a member of class TOutlineNode.
	GetNodeAtIndex (see page 2947)	This is GetNodeAtIndex, a member of class TOutlineNode.
	GetRowOfNode (see page 2948)	This is GetRowOfNode, a member of class TOutlineNode.
	GetTextItem (see page 2948)	This is GetTextItem, a member of class TOutlineNode.
	GetVisibleNode (see page 2949)	This is GetVisibleNode, a member of class TOutlineNode.
	HasAsParent (see page 2949)	This is HasAsParent, a member of class TOutlineNode.
	InsertNode (see page 2949)	This is InsertNode, a member of class TOutlineNode.
	InternalRemove (see page 2949)	This is InternalRemove, a member of class TOutlineNode.
	Remove (see page 2950)	This is Remove, a member of class TOutlineNode.
	WriteNode (see page 2950)	This is the overview for the WriteNode method overload.

Public Methods

	Name	Description
	ChangeLevelBy (see page 2944)	Changes the level of an outline node.
	Collapse (see page 2944)	Collapses an outline item.
	Expand (see page 2945)	Expands an outline item.
	FullExpand (see page 2945)	Expands an outline node and all its descendants.
	GetDisplayWidth (see page 2946)	Returns the distance, in pixels, from the left edge of the outline to the right edge of the outline node.
	getFirstChild (see page 2946)	Returns the Index (see page 2953) value of the first subnode.
	GetLastChild (see page 2947)	Returns the Index (see page 2953) value of the last subnode.
	GetNextChild (see page 2947)	Returns the Index (see page 2953) value of the next child of an outline node.
	GetPrevChild (see page 2948)	Returns the Index (see page 2953) value of the previous child of an outline node.
	MoveTo (see page 2949)	Moves an outline node from one location within the outline to another.

53.1.5.4.1 TOutlineNode.AddNode Method

Pascal

```
function AddNode(Value: TOutlineNode): LongInt;
```

C++

```
__fastcall LongInt AddNode(TOutlineNode Value);
```

Description

This is AddNode, a member of class TOutlineNode.

53.1.5.4.2 TOutlineNode.ChangeLevelBy Method

Changes the level of an outline node.

Pascal

```
procedure ChangeLevelBy(Value: TChangeRange);
```

C++

```
__fastcall ChangeLevelBy(TChangeRange Value);
```

Description

Call ChangeLevelBy to move a node up or down a branch in the outline. Specify a Value parameter value of -1 to move up (toward the root) one level. Specify a Value parameter value of 1 to move down (away from the root) one level.

When moving up one level, an item becomes the next sibling of its former parent. When moving down one level, an item becomes the last child of its former prior sibling.

Nodes at the first level can not be moved up, as they have no parent. The first node in the outline can not be moved down either, as it has no prior sibling.

Note: Before calling ChangeLevelBy, be sure that the node has the appropriate number of parents or that the appropriate prior siblings exist.

See Also

Level (see page 2954)

MoveTo (see page 2949)

Parent (see page 2955)

53.1.5.4.3 TOutlineNode.Collapse Method

Collapses an outline item.

Pascal

```
procedure Collapse;
```

C++

```
__fastcall Collapse();
```

Description

Collapse sets the Expanded (see page 2952) property to false. When an outline node is collapsed, its descendants are hidden and, depending on the OutlineStyle property of the outline in which the node is contained, the plus picture or closed picture is displayed.

Collapse does nothing to nodes that have no descendants.

See Also

Expand (see page 2945)

Expanded (see page 2952)

OutlineStyle (see page 2867)

53.1.5.4.4 TOutlineNode.Expand Method

Expands an outline item.

Pascal

```
procedure Expand;
```

C++

```
__fastcall Expand();
```

Description

Expand sets the Expanded (see page 2952) property to true. When an outline node is expanded, its descendants are displayed and, depending on the OutlineStyle property of the outline in which the node is contained, the minus picture or open picture is displayed.

Expand does nothing to nodes that have no descendants.

Note: Expand only ensures that the immediate children of a node are visible. To expand all descendants of a node, use the FullExpand (see page 2945) method.

See Also

Collapse (see page 2944)

Expanded (see page 2952)

FullExpand (see page 2945)

OutlineStyle (see page 2867)

53.1.5.4.5 TOutlineNode.FullExpand Method

Expands an outline node and all its descendants.

Pascal

```
procedure FullExpand;
```

C++

```
__fastcall FullExpand();
```

Description

Use FullExpand to expand all the nodes in the subtree of the outline rooted at the outline node. FullExpand sets the Expanded (see page 2952) property of the node and all its descendants to true.

See Also

Collapse (see page 2944)

Expand (see page 2945)

Expanded (see page 2952)

53.1.5.4.6 TOutlineNode.GetDataItem Method

Pascal

```
function GetDataItem(Value: TCustomData): LongInt;
```

C++

```
__fastcall LongInt GetDataItem(TCustomData Value);
```

Description

This is GetDataItem, a member of class TOutlineNode.

53.1.5.4.7 TOutlineNode.GetDisplayWidth Method

Returns the distance, in pixels, from the left edge of the outline to the right edge of the outline node.

Pascal

```
function GetDisplayWidth: Integer;
```

C++

```
__fastcall int GetDisplayWidth();
```

Description

Call GetDisplayWidth to determine the length of the line in the outline that contains the outline node. GetDisplayWidth adjusts for the lines and bitmaps specified by the OutlineStyle property of the outline.

See Also

GetNodeDisplayWidth (see page 2852)

OutlineStyle (see page 2867)

53.1.5.4.8 TOutlineNode.getFirstChild Method

Returns the Index (see page 2953) value of the first subnode.

Pascal

```
function getFirstChild: LongInt;
```

C++

```
__fastcall LongInt getFirstChild();
```

Description

Call getFirstChild to locate the first subnode of a node when traversing the outline tree. If the node has no descendants, getFirstChild returns -1.

See Also

GetLastChild (see page 2947)

GetNextChild (see page 2947)

GetPrevChild (see page 2948)

HasItems (see page 2953)

Index (see page 2953)

Parent (see page 2955)

53.1.5.4.9 TOutlineNode.GetLastChild Method

Returns the Index (see page 2953) value of the last subnode.

Pascal

```
function GetLastChild: LongInt;
```

C++

```
__fastcall LongInt GetLastChild();
```

Description

Call GetLastChild to locate the last subnode of a node. If the node has no descendants, GetLastChild returns -1.

See Also

getFirstChild (see page 2946)

GetNextChild (see page 2947)

GetPrevChild (see page 2948)

HasItems (see page 2953)

Index (see page 2953)

53.1.5.4.10 TOutlineNode.GetNextChild Method

Returns the Index (see page 2953) value of the next child of an outline node.

Pascal

```
function GetNextChild(Value: LongInt): LongInt;
```

C++

```
__fastcall LongInt GetNextChild(LongInt Value);
```

Description

Call GetNextChild to obtain the index of the next immediate descendant of a node, following the child node with the index given by the Value parameter. If the child node indicated by Value has no next sibling, or if the child node indicated by Value is not an immediate descendant of the outline node, GetNextChild returns -1.

Index (see page 2953) values reflect a depth-first traversal of the outline. Thus, the node returned by GetNextChild is not simply the node with an index of Value+1, unless the node with index Value has no descendants. GetNextChild allows a breadth-first traversal of the outline.

See Also

getFirstChild (see page 2946)

GetLastChild (see page 2947)

GetPrevChild (see page 2948)

Index (see page 2953)

53.1.5.4.11 TOutlineNode.GetNodeAtIndex Method

Pascal

```
function GetNodeAtIndex(TargetIndex: LongInt): TOutlineNode;
```

C++

```
__fastcall TOutlineNode GetNodeAtIndex(LongInt TargetIndex);
```

Description

This is GetNodeAtIndex, a member of class TOutlineNode.

53.1.5.4.12 TOutlineNode.GetPrevChild Method

Returns the Index (see page 2953) value of the previous child of an outline node.

Pascal

```
function GetPrevChild(Value: LongInt): LongInt;
```

C++

```
__fastcall LongInt GetPrevChild(LongInt Value);
```

Description

Call GetPrevChild to obtain the index of the previous immediate descendant of a node, preceding the child node with the index given by the Value parameter. If the child node indicated by Value has no prior sibling, or if the child node indicated by Value is not an immediate descendant of the outline node, GetPrevChild returns -1.

Index (see page 2953) values reflect a depth-first traversal of the outline. Thus, the previous sibling of the child node indicated by Value is not simply one less than Value, unless that previous sibling has no descendants.

See Also

getFirstChild (see page 2946)

GetLastChild (see page 2947)

GetNextChild (see page 2947)

Index (see page 2953)

53.1.5.4.13 TOutlineNode.GetRowOfNode Method

Pascal

```
function GetRowOfNode(TargetNode: TOutlineNode; var RowCount: Longint): Boolean;
```

C++

```
__fastcall Boolean GetRowOfNode(TOutlineNode TargetNode, Longint RowCount);
```

Description

This is GetRowOfNode, a member of class TOutlineNode.

53.1.5.4.14 TOutlineNode.GetTextItem Method

Pascal

```
function GetTextItem(const Value: string): LongInt;
```

C++

```
__fastcall LongInt GetTextItem(const UnicodeString Value);
```

Description

This is GetTextItem, a member of class TOutlineNode.

53.1.5.4.15 TOutlineNode.GetVisibleNode Method

Pascal

```
function GetVisibleNode(TargetCount: LongInt): TOutlineNode;
```

C++

```
__fastcall TOutlineNode GetVisibleNode(LongInt TargetCount);
```

Description

This is GetVisibleNode, a member of class TOutlineNode.

53.1.5.4.16 TOutlineNode.HasAsParent Method

Pascal

```
function HasAsParent(Value: TOutlineNode): Boolean;
```

C++

```
__fastcall Boolean HasAsParent(TOutlineNode Value);
```

Description

This is HasAsParent, a member of class TOutlineNode.

53.1.5.4.17 TOutlineNode.InsertNode Method

Pascal

```
function InsertNode(Index: LongInt; Value: TOutlineNode): LongInt;
```

C++

```
__fastcall LongInt InsertNode(LongInt Index, TOutlineNode Value);
```

Description

This is InsertNode, a member of class TOutlineNode.

53.1.5.4.18 TOutlineNode.InternalRemove Method

Pascal

```
procedure InternalRemove(Value: TOutlineNode; Index: Integer);
```

C++

```
__fastcall InternalRemove(TOutlineNode Value, int Index);
```

Description

This is InternalRemove, a member of class TOutlineNode.

53.1.5.4.19 TOutlineNode.MoveTo Method

Moves an outline node from one location within the outline to another.

Pascal

```
procedure MoveTo(Destination: LongInt; AttachMode: TAttachMode);
```

C++

```
__fastcall MoveTo(LongInt Destination, TAttachMode AttachMode);
```

Description

Call MoveTo to reposition the node to the position indicated by the Destination parameter. Destination is the Index (see page 2953) value of another outline node. The AttachMode parameter specifies how to attach the moved node to the destination position.

Note: To change only the level of a node, use the ChangeLevelBy (see page 2944) method.

See Also

ChangeLevelBy (see page 2944)

Index (see page 2953)

Add (see page 2842)

AddChild (see page 2843)

Insert (see page 2853)

53.1.5.4.20 TOutlineNode.Remove Method

Pascal

```
procedure Remove(Value: TOutlineNode);
```

C++

```
__fastcall Remove(TOutlineNode Value);
```

Description

This is Remove, a member of class TOutlineNode.

53.1.5.4.21 WriteNode Method

This is the overview for the WriteNode method overload.

Overload List

	Name	Description
	TOutlineNode.WriteNode (PChar, TStream) (see page 2950)	This is WriteNode, a member of class TOutlineNode.
	TOutlineNode.WriteNode (TStream, TEncoding) (see page 2951)	This is WriteNode, a member of class TOutlineNode.

53.1.5.4.21.1 TOutlineNode.WriteNode Method (PChar, TStream)

Pascal

```
procedure WriteNode(Buffer: PChar; Stream: TStream); overload; deprecated;
```

C++

```
__fastcall WriteNode(const char * Buffer, TStream Stream);
```

Description

This is WriteNode, a member of class TOutlineNode.

53.1.5.4.21.2 TOutlineNode.WriteNode Method (TStream, TEncoding)

Pascal

```
procedure WriteNode(Stream: TStream; Encoding: TEncoding); overload;
```

C++

```
__fastcall WriteNode(TStream Stream, TEncoding Encoding);
```

Description

This is WriteNode, a member of class TOutlineNode.

53.1.5.5 TOutlineNode Properties

The properties of the TOutlineNode class are listed here.

Protected Properties

	Name	Description
	ExpandCount (see page 2952)	This is ExpandCount, a member of class TOutlineNode.
	Items (see page 2954)	This is Items, a member of class TOutlineNode.
	List (see page 2955)	This is List, a member of class TOutlineNode.
	Outline (see page 2955)	This is Outline, a member of class TOutlineNode.

Public Properties

	Name	Description
	Data (see page 2951)	Points to application-specific data associated with the node.
	Expanded (see page 2952)	Determines whether descendants of the outline node are displayed.
	FullPath (see page 2952)	Describes an outline node, using its text and the text of all ancestor nodes in the outline.
	HasItems (see page 2953)	Indicates whether the outline node has any subnodes.
	Index (see page 2953)	Identifies the node in the Items (see page 2954) array maintained by the outline.
	IsVisible (see page 2954)	Indicates whether the outline node is visible in the outline.
	Level (see page 2954)	Indicates the level of indentation of a node within the outline.
	Parent (see page 2955)	Identifies the immediate ancestor of the node.
	Text (see page 2955)	Specifies the string that identifies the outline node.
	TopItem (see page 2956)	Indicates which outline node is the topmost ancestor of a node.

53.1.5.5.1 TOutlineNode.Data Property

Points to application-specific data associated with the node.

Pascal

```
property Data: TCustomData;
```

C++

```
__property TCustomData Data;
```

Description

Use Data to associate an arbitrary value with a node in an outline. For example, if the outline represents the table of contents to an electronic document, Data could point to an object that provides jump information for accessing the section of the document named by the node.

The Data property is usually described by the Text (see page 2955) property, which is displayed in the rendered image of the node.

See Also

Text (see page 2955)

53.1.5.5.2 TOutlineNode.ExpandCount Property

Pascal

```
property ExpandCount: LongInt;
```

C++

```
__property LongInt ExpandCount;
```

Description

This is ExpandCount, a member of class TOutlineNode.

53.1.5.5.3 TOutlineNode.Expanded Property

Determines whether descendants of the outline node are displayed.

Pascal

```
property Expanded: Boolean;
```

C++

```
__property Boolean Expanded;
```

Description

Set Expanded to true to display all subnodes of a node. When an outline node is expanded, its descendants are displayed and, depending on the OutlineStyle property of the outline in which the node is contained, the minus picture or open picture is displayed.

Set Expanded to false to collapse the node. When an outline node is collapsed, its descendants are hidden and, depending on the OutlineStyle property of the outline in which the node is contained, the plus picture or closed picture is displayed.

Expanded is not meaningful for nodes that have no descendants.

See Also

HasItems (see page 2953)

OutlineStyle (see page 2867)

53.1.5.5.4 TOutlineNode.FullPath Property

Describes an outline node, using its text and the text of all ancestor nodes in the outline.

Pascal

```
property FullPath: string;
```

C++

```
__property UnicodeString FullPath;
```

Description

Read FullPath to obtain a description of the entire branch of the outline leading to the outline node. FullPath consists of the values of the Text (see page 2955) properties of all outline items, starting with the text of the node at the first level and traversing the outline down to the outline node. The string specified as the ItemSeparator property of the outline separates the

text for each node in the path.

See Also

Level (see page 2954)

Parent (see page 2955)

ItemSeparator (see page 2865)

Text (see page 2955)

TopItem (see page 2956)

53.1.5.5.5 TOutlineNode.HasItems Property

Indicates whether the outline node has any subnodes.

Pascal

```
property HasItems: Boolean;
```

C++

```
__property Boolean HasItems;
```

Description

Read HasItem to determine whether the outline node is a leaf node. Depending on the OutlineStyle property of the outline that contains the node, leaf nodes may appear with the leaf bitmap. If the node is not a leaf, subnodes appear below and indented from the node when it is expanded.

HasItem returns true if the node has any descendants, even if the node is collapsed and the descendants are not visible. HasItem returns false if the node is a leaf node.

See Also

Expanded (see page 2952)

Parent (see page 2955)

OutlineStyle (see page 2867)

53.1.5.5.6 TOutlineNode.Index Property

Identifies the node in the Items (see page 2954) array maintained by the outline.

Pascal

```
property Index: LongInt;
```

C++

```
__property LongInt Index;
```

Description

Read Index to determine the row of the outline where the node appears. The first node has an Index value of 1, and subsequent nodes are indexed sequentially. If a node has subnodes, the Index value of the parent node is one less than the Index value of its first descendant.

When nodes in the outline are added, moved, or deleted, the values of the Index properties of all subsequent nodes must be recalculated to be valid. Sometimes, the recalculation of indexes is deferred until all changes to the nodes are complete. To ensure that the value of Index is valid, call the SetUpdateState method of the outline that contains the node.

See Also

Items (see page 2865)

SetUpdateState (see page 2861)

53.1.5.5.7 TOutlineNode.IsVisible Property

Indicates whether the outline node is visible in the outline.

Pascal

```
property IsVisible: Boolean;
```

C++

```
__property Boolean IsVisible;
```

Description

Read IsVisible to determine whether the node is part of the current image of the outline. A node is visible if it is on level 1 or if all its parents are expanded.

Note: Do not confuse IsVisible with the Visible property of TControl objects. Outline (see page 2955) nodes are not controls, but merely helper objects that describe nodes for an outline control. IsVisible indicates whether a node is visible because its parents are expanded or collapsed. It does not provide information about whether the node has been scrolled out of view, or even if the outline is not visible.

See Also

Expanded (see page 2952)

Visible

53.1.5.5.8 TOutlineNode.Items Property

Pascal

```
property Items [Index: LongInt]: TOutlineNode;
```

C++

```
__property TOutlineNode Items[LongInt Index];
```

Description

This is Items, a member of class TOutlineNode.

53.1.5.5.9 TOutlineNode.Level Property

Indicates the level of indentation of a node within the outline.

Pascal

```
property Level: Cardinal;
```

C++

```
__property unsigned Level;
```

Description

The value of Level is 1 for nodes that directly descend from the root of the outline (the top level). The value of Level is 2 for their children, and so on.

See Also

ChangeLevelBy (see page 2944)

Parent (see page 2955)

TopItem (see page 2956)

53.1.5.5.10 TOutlineNode.List Property

Pascal

```
property List: TList;
```

C++

```
__property TList List;
```

Description

This is List, a member of class TOutlineNode.

53.1.5.5.11 TOutlineNode.Outline Property

Pascal

```
property Outline: TCustomOutline;
```

C++

```
__property TCustomOutline Outline;
```

Description

This is Outline, a member of class TOutlineNode.

53.1.5.5.12 TOutlineNode.Parent Property

Identifies the immediate ancestor of the node.

Pascal

```
property Parent: TOutlineNode;
```

C++

```
__property TOutlineNode Parent;
```

Description

Read Parent to find the parent of an outline node. A parent node is one level higher and contains the child node as a subnode.

See Also

Level (see page 2954)

TopItem (see page 2956)

53.1.5.5.13 TOutlineNode.Text Property

Specifies the string that identifies the outline node.

Pascal

```
property Text: string;
```

C++

```
__property UnicodeString Text;
```

Description

Use Text to determine the string that is displayed in the outline to represent the node.

See Also

FullPath (see page 2952)

53.1.5.5.14 TOutlineNode.TopItem Property

Indicates which outline node is the topmost ancestor of a node.

Pascal

```
property TopItem: Longint;
```

C++

```
__property Longint TopItem;
```

Description

Read TopItem to determine the first node in the branch of the outline tree that contains the outline node. The TopItem property specifies the Index (see page 2953) value of the level 1 ancestor of the outline node. For a node on level 1, TopItem has the same value as the Index (see page 2953) property.

See Also

Index (see page 2953)

Level (see page 2954)

Parent (see page 2955)

53.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TAttachMode (see page 2957)	TAttachMode indicates how an outline node should be inserted into an outline.
	TOutlineBitmap (see page 2958)	This is record Outline.TOutlineBitmap.
	TOutlineNodeCompare (see page 2958)	This is record Outline.TOutlineNodeCompare.
	TOutlineOption (see page 2959)	TOutlineOptions (see page 2964) represents the style in which an outline is drawn.
	TOutlineStyle (see page 2959)	TOutlineStyle determines how the nodes of an outline are drawn.
	TOutlineType (see page 2960)	TOutlineType indicates whether an outline is owner-drawn.

53.2.1 Outline.TAttachMode Enumeration

TAttachMode indicates how an outline node should be inserted into an outline.

Pascal

```
TAttachMode = (
  oaAdd,
  oaAddChild,
  oaInsert
);
```

C++

```
enum TAttachMode {
  oaAdd,
  oaAddChild,
  oaInsert
};
```

File

Outline

Description

TAttachMode is used by methods that move outline nodes within an outline. It includes the following values:

Value	Description
oaAdd	The item is attached as if added with the Add method. The moved item becomes the last sibling of the item specified by the Destination parameter, sharing the same parent as the Destination item.
oaAddChild	The item is attached as if added with the AddChild method. The moved item becomes the last child of the item specified by the Destination parameter. The Destination item becomes the parent of the moved item.
oaInsert	The item is attached as if inserted with the Insert method. The moved item replaces the Destination item in the outline, while the Destination item and all other following items are moved down one row.

See Also

TCustomOutline (see page 2838)

TOutlineNode (see page 2940)

53.2.2 Outline.TOutlineBitmap Enumeration

Pascal

```
TOutlineBitmap = (
  obPlus,
  obMinus,
  obOpen,
  obClose,
  obLeaf
);
```

C++

```
enum TOutlineBitmap {
  obPlus,
  obMinus,
  obOpen,
  obClose,
  obLeaf
};
```

File

Outline

Description

This is record Outline.TOutlineBitmap.

53.2.3 Outline.TOutlineNodeCompare Enumeration

Pascal

```
TOutlineNodeCompare = (
  ocLess,
  ocSame,
  ocGreater,
  ocInvalid
);
```

C++

```
enum TOutlineNodeCompare {
  ocLess,
  ocSame,
  ocGreater,
  ocInvalid
};
```

File

Outline

Description

This is record Outline.TOutlineNodeCompare.

53.2.4 Outline.TOutlineOption Enumeration

TOutlineOptions (see page 2964) represents the style in which an outline is drawn.

Pascal

```
TOutlineOption = (
  ooDrawTreeRoot,
  ooDrawFocusRect,
  ooStretchBitmaps
);
```

C++

```
enum TOutlineOption {
  ooDrawTreeRoot,
  ooDrawFocusRect,
  ooStretchBitmaps
};
```

File

Outline

Description

TOutlineOptions (see page 2964) is a set of TOutlineOption values. Together they influence the way an outline is drawn. Each TOutlineOption value is described below:

Value	Meaning
ooDrawTreeRoot	The first item (Index value of 1) is connected to the root item by the outline tree. This means that the tree will extend from the top of the outline to all the first level items. Without ooDrawTreeRoot, all first level items appear leftmost in the outline, not connected by the tree.
ooDrawFocusRect	The outline draws a focus rectangle around the selected item.
ooStretchBitmaps	The outline stretches the standard bitmaps (PictureLeaf, PictureOpen, PictureClosed, PicturePlus, PictureMinus) to fit in the size of the item, determined by the size of the Font of the Text. Without ooStretchBitmaps, the bitmaps will be cropped if larger than the height of the item text, or won't fill up the entire item space if smaller than the text.

See Also

TCustomOutline (see page 2838)

53.2.5 Outline.TOutlineStyle Enumeration

TOutlineStyle determines how the nodes of an outline are drawn.

Pascal

```
TOutlineStyle = (
  osText,
  osPlusMinusText,
  osPictureText,
  osPlusMinusPictureText,
  osTreeText,
  osTreePictureText
);
```

C++

```
enum TOutlineStyle {
  osText,
  osPlusMinusText,
  osPictureText,
  osPlusMinusPictureText,
  osTreeText,
  osTreePictureText
};
```

File

Outline

Description

TOutlineStyle is the type of TCustomOutline (see page 2838).::OutlineStyle. It includes the following values:

Value	Meaning
osText	Only the text of nodes is drawn.
osPlusMinusText	In addition to the text, each node displays plus picture (when it can be expanded) or minus picture (when it can be collapsed).
osPictureText	In addition to the text, each node displays the open picture (when it is expanded), closed picture (when it is collapsed), or leaf picture (when it has no subnodes).
osPlusMinusPictureText	In addition to the text, each node displays the plus picture or minus picture (as in osPlusMinusText), and the open, closed or leaf picture (as in osPictureText).
osTreeText	Visible lines connect the outline nodes.
osTreePictureText	In addition to the text, each node displays the open picture (when it is expanded), closed picture (when it is collapsed), or leaf picture (when it has no subnodes). Visible lines connect the outline nodes.

See Also

TCustomOutline (see page 2838)

53.2.6 Outline.TOutlineType Enumeration

TOutlineType indicates whether an outline is owner-drawn.

Pascal

```
TOutlineType = (
  otStandard,
  otOwnerDraw
);
```

C++

```
enum TOutlineType {
  otStandard,
  otOwnerDraw
};
```

File

Outline

Description

TOutlineType determines whether an outline is drawn automatically or by user code in an OnDrawItem event.

Value	Meaning
otStandardItems	Items are drawn automatically.
otOwnerDraw	Items are drawn on the Canvas by application code.

See Also

TCustomOutline (see page 2838)

53.3 Types

The following table lists types in this documentation.

Types

Name	Description
EOutlineChange (see page 2962)	EOutlineChange is the type of event handlers that respond when the nodes in an outline control change.
TBitmapArray (see page 2962)	This is type Outline.TBitmapArray.
TBitmapArrayRange (see page 2963)	This is type Outline.TBitmapArrayRange.
TChangeRange (see page 2963)	TChangeRange indicates the direction in which a node is moved.
TOutlineBitmaps (see page 2963)	This is type Outline.TOutlineBitmaps.
TOutlineOptions (see page 2964)	TOutlineOptions represents the style in which an outline is drawn.

53.3.1 Outline.EOutlineChange Type

EOutlineChange is the type of event handlers that respond when the nodes in an outline control change.

Pascal

```
EOutlineChange = procedure (Sender: TObject; Index: LongInt) of object;
```

C++

```
(Sender: TObject; Index: LongInt) ( EOutlineChange)();
```

File

Outline

Description

Sender is the outline control whose node just changed.

Index is the index of the node that just changed.

53.3.2 Outline.TBitmapArray Type

Pascal

```
TBitmapArray = array[TBitmapArrayRange] of TBitmap;
```

C++

```
array[TBitmapArrayRange] of TBitmap TBitmapArray;
```

File

Outline

Description

This is type Outline.TBitmapArray.

53.3.3 Outline.TBitmapArrayRange Type

Pascal

```
TBitmapArrayRange = 0..4;
```

C++

```
0..4 TBitmapArrayRange;
```

File

Outline

Description

This is type Outline.TBitmapArrayRange.

53.3.4 Outline.TChangeRange Type

TChangeRange indicates the direction in which a node is moved.

Pascal

```
TChangeRange = -1..1;
```

C++

```
-1..1 TChangeRange;
```

File

Outline

Description

TChangeRange is the type of the Value parameter of an outline node's ChangeLevelBy method. It can take one of the following values:

Value	Meaning
-1	Move the node toward the root by one level.
1	Move the node toward the leaf by one level.

See Also

TOutlineNode (see page 2940)

53.3.5 Outline.TOutlineBitmaps Type

Pascal

```
TOutlineBitmaps = set of TOutlineBitmap;
```

C++

```
TOutlineBitmap TOutlineBitmaps;
```

File

Outline

Description

This is type Outline.TOutlineBitmaps.

53.3.6 Outline.TOutlineOptions Type

TOutlineOptions represents the style in which an outline is drawn.

Pascal

```
TOutlineOptions = set of TOutlineOption;
```

C++

```
TOutlineOption TOutlineOptions;
```

File

Outline

Description

TOutlineOptions is a set of TOutlineOption (see page 2959) values. Together they influence the way an outline is drawn. Each TOutlineOption (see page 2959) value is described below:

Value	Meaning
ooDrawTreeRoot	The first item (Index value of 1) is connected to the root item by the outline tree. This means that the tree will extend from the top of the outline to all the first level items. Without ooDrawTreeRoot, all first level items appear leftmost in the outline, not connected by the tree.
ooDrawFocusRect	The outline draws a focus rectangle around the selected item.
ooStretchBitmaps	The outline stretches the standard bitmaps (PictureLeaf, PictureOpen, PictureClosed, PicturePlus, PictureMinus) to fit in the size of the item, determined by the size of the Font of the Text. Without ooStretchBitmaps, the bitmaps will be cropped if larger than the height of the item text, or won't fill up the entire item space if smaller than the text.

See Also

TCustomOutline (see page 2838)

54 PagItems Namespace

This is namespace PagItems.

Topics

Name	Description
Classes (see page 2966)	The following table lists classes in this documentation.

Classes

	Name	Description
	TBasePageItemsProducer (see page 2966)	TBasePageItemsProducer produces a string of HTML commands based on an input template.
	TPageItemsProducer (see page 2970)	TPageItemsProducer maintains a list of Web item components for producing a string of HTML commands based on an input template.

54.1 Classes

The following table lists classes in this documentation.

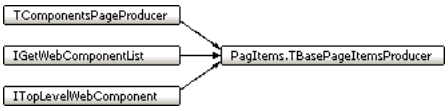
Classes

	Name	Description
	TBasePageItemsProducer (see page 2966)	TBasePageItemsProducer produces a string of HTML commands based on an input template.
	TPageItemsProducer (see page 2970)	TPageItemsProducer maintains a list of Web item components for producing a string of HTML commands based on an input template.

54.1.1 TBasePageItemsProducer Class

TBasePageItemsProducer produces a string of HTML commands based on an input template.

Class Hierarchy



Pascal

```
TBasePageItemsProducer = class(TComponentsPageProducer, IGetWebComponentList,
ITopLevelWebComponent);
```

C++

```
class TBasePageItemsProducer : public TComponentsPageProducer, public IGetWebComponentList,
public ITopLevelWebComponent;
```

File

PagItems

Description

Use TBasePageItemsProducer to convert an HTML template that includes content generated by other components into a string of HTML commands that can be interpreted by a client application such as a Web browser. HTML templates include HTML commands and HTML-transparent tags that are replaced with customized content by the components page producer in an OnHTMLTag event handler.

TBasePageItemsProducer differs from other page producer components in that it replaces special HTML-transparent tags of the format

```
<#COMPONENT Name=WebContentComponent>
```

with the HTML produced by the components named WebContentComponent. In addition, TBasePageItemsProducer stores a default value for WebContentComponent as the value of its WebPageItems (see page 2969) property.

See Also

IWebContent

TPageItemsProducer (see page 2970)

TInetXPageProducer (see page 2285)

TAdapterPageProducer

54.1.1.1 TBasePageItemsProducer Members

The following tables list the members exposed by TBasePageItemsProducer.

Protected Methods

	Name	Description
	GetChildren (see page 2968)	Enumerates all child components.
	GetComponentList (see page 2968)	This is GetComponentList, a member of class TBasePageItemsProducer.
	GetDefaultComponentList (see page 2969)	This is GetDefaultComponentList, a member of class TBasePageItemsProducer.
	SetChildOrder (see page 2969)	Sets the visual ordering of a child control.

Public Methods

	Name	Description
	Create (see page 2967)	Creates an instance of a TBasePageItemsProducer (see page 2966).
	Destroy (see page 2967)	Destroys the instance of the TBasePageItemsProducer (see page 2966).

Public Properties

	Name	Description
	WebPageItems (see page 2969)	Maintains a list of components that generate HTML content for the producer.

54.1.1.2 TBasePageItemsProducer.Create Constructor

Creates an instance of a TBasePageItemsProducer (see page 2966).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TBasePageItemsProducer(TComponent * AOwner);
```

Description

Call Create to instantiate a TBasePageItemsProducer (see page 2966) component at runtime if it was not created at design time.

AOwner is the component, typically the Web module, that becomes the value of this page producer's Owner property. The Owner is responsible for freeing the TBasePageItemsProducer (see page 2966) instance.

See Also

Owner

54.1.1.3 TBasePageItemsProducer.Destroy Destructor

Destroys the instance of the TBasePageItemsProducer (see page 2966).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TBasePageItemsProducer();
```

Description

Do not call Destroy directly. Instead call Free, which checks that the TBasePageItemsProducer (see page 2966) reference is not nil before calling Destroy.

See Also

Create (see page 2967)

System::TObject::Free

54.1.1.4 TBasePageItemsProducer Methods

The methods of the TBasePageItemsProducer class are listed here.

Protected Methods

	Name	Description
	GetChildren (see page 2968)	Enumerates all child components.
	GetComponentList (see page 2968)	This is GetComponentList, a member of class TBasePageItemsProducer.
	GetDefaultComponentList (see page 2969)	This is GetDefaultComponentList, a member of class TBasePageItemsProducer.
	SetChildOrder (see page 2969)	Sets the visual ordering of a child control.

54.1.1.4.1 TBasePageItemsProducer.GetChildren Method

Enumerates all child components.

Pascal

```
procedure GetChildren(Proc: TGetChildProc; Root: TComponent); override;
```

C++

```
virtual __fastcall GetChildren(TGetChildProc Proc, TComponent * Root);
```

Description

GetChildren is used internally in the component streaming system. It is not necessary to call it directly.

GetChildren expects a TGetChildProc routine that receives all enumerated components.

`Root` represents the owner of the components that will be enumerated.

This method does nothing in TComponent and is expected to be overridden in descendants.

See Also

TGetChildProc

GetParentComponent

54.1.1.4.2 TBasePageItemsProducer.GetComponentList Method

Pascal

```
function GetComponentList: TObject;
```

C++

```
__fastcall TObject * GetComponentList();
```

Description

This is GetComponentList, a member of class TBasePageItemsProducer.

54.1.1.4.3 TBasePageItemsProducer.GetDefaultComponentList Method

Pascal

```
function GetDefaultComponentList: TObject;
```

C++

```
__fastcall TObject * GetDefaultComponentList();
```

Description

This is GetDefaultComponentList, a member of class TBasePageItemsProducer.

54.1.1.4.4 TBasePageItemsProducer.SetChildOrder Method

Sets the visual ordering of a child control.

Pascal

```
procedure SetChildOrder(Component: TComponent; Order: Integer); override;
```

C++

```
virtual __fastcall SetChildOrder(TComponent * Component, int Order);
```

Description

SetChildOrder is used internally in the component streaming system. It is not necessary to call it directly.

Derived classes override this method to actually implement the ordering. TWinControl uses this ordering information to Z-align the controls.

See Also

SetZOrder

54.1.1.5 TBasePageItemsProducer Properties

The properties of the TBasePageItemsProducer class are listed here.

Public Properties

	Name	Description
	WebPageItems (see page 2969)	Maintains a list of components that generate HTML content for the producer.

54.1.1.5.1 TBasePageItemsProducer.WebPageItems Property

Maintains a list of components that generate HTML content for the producer.

Pascal

```
property WebPageItems: TWebComponentList;
```

C++

```
__property TWebComponentList WebPageItems;
```

Description

WebPageItems lists the components that generate content for the page items producer. Each of these components supports the IWebContent interface, and can contain its own list of Web items that help it generate its content.

The type of items that can be added to the WebPageItems list are restricted by a protected method that is abstract or, in C++ terminology, pure virtual in TBasePageItemsProducer (see page 2966). Descendant classes implement this protected method to indicate what types of items can be added to the list.

At design time, use the Web page editor to add components to the WebPageItems property.

Note: Typically, WebPageItems lists all the components that generate content for the page items producer, in the order that the generated content appears in the final HTML document. However, the page items producer can transform COMPONENT tags in its template using components that are not in this list, as long as they have the same owner as the page items producer..

See Also

IWebContent

Creating Web Pages with an InternetExpress Page Producer

54.1.2 TPageItemsProducer Class

TPageItemsProducer maintains a list of Web item components for producing a string of HTML commands based on an input template.

Class Hierarchy

Pascal

```
TPageItemsProducer = class(TBasePageItemsProducer);
```

C++

```
class TPageItemsProducer : public TBasePageItemsProducer;
```

File

PagItems

Description

Use TPageItemsProducer as the base class for components that convert an HTML template that includes content generated by other components into a string of HTML commands that can be interpreted by a client application such as a Web browser. HTML templates include HTML commands and HTML-transparent tags that are replaced with customized content by the page items producer in an OnHTMLTag event handler.

Page item producers differ from other page producer components in that they replace special HTML-transparent tags of the format

```
<#COMPONENT Name=WebContentComponent>
```

with the HTML produced by the component named WebContentComponent. WebContentComponent must have the same Owner as the page item producer and support the IWebContent interface, which the page item producer uses to obtain the replacement HTML for the COMPONENT tag.

TPageItemsProducer introduces the WebPageItems (see page 2969) property to maintain a hierarchical list of these Web content components, where each component produces content for its parent component.

See Also

IWebContent

TInetXPageProducer (see page 2285)

54.1.2.1 TPageItemsProducer Members

The following tables list the members exposed by TPageItemsProducer.

Protected Methods

	Name	Description
	GetChildren (see page 2968)	Enumerates all child components.
	GetComponentList (see page 2968)	This is GetComponentList, a member of class TBasePageItemsProducer.
	GetDefaultComponentList (see page 2969)	This is GetDefaultComponentList, a member of class TBasePageItemsProducer.
	GetTemplateStream (see page 2972)	This is GetTemplateStream, a member of class TPageItemsProducer.
	SetChildOrder (see page 2969)	Sets the visual ordering of a child control.

Public Methods

	Name	Description
	Create (see page 2971)	Creates and initializes an instance of TPageItemsProducer (see page 2970).
	Destroy (see page 2971)	Frees an instance of TPageItemsProducer (see page 2970).

Public Properties

	Name	Description
	HTMLDoc (see page 2972)	Specifies the HTML template used to generate the value returned by the Content method.
	HTMLFile (see page 2973)	Specifies the name of a file that contains the HTML template used to generate the value returned by the Content method.
	WebPageItems (see page 2969)	Maintains a list of components that generate HTML content for the producer.

54.1.2.2 TPageItemsProducer.Create Constructor

Creates and initializes an instance of TPageItemsProducer (see page 2970).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TPageItemsProducer(TComponent * AOwner);
```

Description

Call Create to instantiate a page items producer at runtime.

54.1.2.3 TPageItemsProducer.Destroy Destructor

Frees an instance of TPageItemsProducer (see page 2970).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TPageItemsProducer();
```

Description

Do not call Destroy directly. Instead, call the Free method, which checks whether the TPageItemsProducer (see page 2970) reference is not nil before calling Destroy.

See Also

System::TObject::Free

54.1.2.4 TPageItemsProducer Methods

The methods of the TPageItemsProducer class are listed here.

Protected Methods

	Name	Description
	GetTemplateStream (see page 2972)	This is GetTemplateStream, a member of class TPageItemsProducer.

54.1.2.4.1 TPageItemsProducer.GetTemplateStream Method

Pascal

```
function GetTemplateStream(out AOwned: Boolean): TStream; override;
```

C++

```
virtual __fastcall TStream GetTemplateStream(Boolean AOwned);
```

Description

This is GetTemplateStream, a member of class TPageItemsProducer.

54.1.2.5 TPageItemsProducer Properties

The properties of the TPageItemsProducer class are listed here.

Public Properties

	Name	Description
	HTMLDoc (see page 2972)	Specifies the HTML template used to generate the value returned by the Content method.
	HTMLFile (see page 2973)	Specifies the name of a file that contains the HTML template used to generate the value returned by the Content method.

54.1.2.5.1 TPageItemsProducer.HTMLDoc Property

Specifies the HTML template used to generate the value returned by the Content method.

Pascal

```
property HTMLDoc: TStrings;
```

C++

```
__property TStrings HTMLDoc;
```

Description

Set HTMLDoc to a set of strings that make up the HTML template. Each string is a sequence of one or more HTML commands or HTML-transparent tags. An HTML-transparent tag has the form

```
<#TagName Param1=Value1 Param2=Value2 ...>
```

A pound sign (#) immediately follows the opening angle bracket (<) with no spaces separating it from the angle bracket. The pound sign identifies the string to the page producer as an HTML-transparent tag. The tag name immediately follows the pound sign with no spaces separating it from the pound sign. The page producer converts the entire HTML-transparent tag into true HTML commands based on the value of the tag name. Following the tag name, the HTML-transparent tag can optionally include parameters that specify details of the conversion to be performed. Each parameter is of the form ParamName=Value, where there is no space between the parameter name, the equals symbol (=) and the value. Each parameter is delimited by white space.

The angle brackets (< >) make the tag transparent to HTML browsers that do not recognize the #TagName construct.

Note: Setting HTMLDoc clears the HTMLFile (see page 2973) property.

See Also

Content

HTMLFile (see page 2973)

54.1.2.5.2 TPageItemsProducer.HTMLFile Property

Specifies the name of a file that contains the HTML template used to generate the value returned by the Content method.

Pascal

```
property HTMLFile: TFileName;
```

C++

```
__property TFileName HTMLFile;
```

Description

Set HTMLFile to the name of the file that contains an HTML template. The HTML template is a sequence of HTML commands and HTML-transparent tags. An HTML-transparent tag has the form

```
<#TagName Param1=Value1 Param2=Value2 ...>
```

A pound sign (#) immediately follows the opening angle bracket (<) with no spaces separating it from the angle bracket. The pound sign identifies the string to the page producer as an HTML-transparent tag. The tag name immediately follows the pound sign with no spaces separating it from the pound sign. The page producer converts the entire HTML-transparent tag into true HTML commands based on the value of the tag name. Following the tag name, the HTML-transparent tag can optionally include parameters that specify details of the conversion to be performed. Each parameter is of the form ParamName=Value, where there is no space between the parameter name, the equals symbol (=) and the value. Each parameter is delimited by white space.

The angle brackets (< >) make the tag transparent to HTML browsers that do not recognize the #TagName construct.

Note: Setting HTMLFile clears the HTMLDoc (see page 2972) property.

Note: When distributing applications that contain page producers which use the HTMLFile property, be sure to distribute the file along with the application. Also, be sure that the value of HTMLFile does not include a hard-coded path name that will not be available on the user's disk.

See Also

Content

HTMLDoc (see page 2972)

55 PlatformDefaultStyleActnCtrls Namespace

This is namespace PlatformDefaultStyleActnCtrls.

Topics

Name	Description
Classes (see page 2976)	The following table lists classes in this documentation.
Variables (see page 2979)	The following table lists variables in this documentation.

Classes

	Name	Description
	TPlatformDefaultStyleActionBars (see page 2976)	This is class PlatformDefaultStyleActnCtrls.TPlatformDefaultStyleActionBars.

Variables

Name	Description
PlatformDefaultStyle (see page 2979)	This is variable PlatformDefaultStyleActnCtrls.PlatformDefaultStyle.

55.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TPlatformDefaultStyleActionBars (see page 2976)	This is class PlatformDefaultStyleActnCtrls.TPlatformDefaultStyleActionBars.

55.1.1 TPlatformDefaultStyleActionBars Class

Class Hierarchy

Pascal

```
TPlatformDefaultStyleActionBars = class(TActionBarStyleEx);
```

C++

```
class TPlatformDefaultStyleActionBars : public TActionBarStyleEx;
```

File

PlatformDefaultStyleActnCtrls

Description

This is class PlatformDefaultStyleActnCtrls.TPlatformDefaultStyleActionBars.

55.1.1.1 TPlatformDefaultStyleActionBars Members

The following tables list the members exposed by TPlatformDefaultStyleActionBars.

Public Methods

	Name	Description
	GetAddRemoveItemClass (see page 2977)	Returns the base class for ActionBand menus for customizing action bands.
	GetColorMapClass (see page 2977)	Returns the class type of the colormap for an action bar.
	GetControlClass (see page 2977)	Returns the control class type on an action bar item.
	GetPopupClass (see page 2978)	Returns the popup class added to a menu action band.
	GetScrollBtnClass (see page 2978)	This is GetScrollBtnClass, a member of class TPlatformDefaultStyleActionBars.
	GetStyleName (see page 2978)	Returns the style associated with action manager.

55.1.1.2 TPlatformDefaultStyleActionBars Methods

The methods of the TPlatformDefaultStyleActionBars class are listed here.

Public Methods

	Name	Description
	GetAddRemoveItemClass (see page 2977)	Returns the base class for ActionBand menus for customizing action bands.

	GetColorMapClass (see page 2977)	Returns the class type of the colormap for an action bar.
	GetControlClass (see page 2977)	Returns the control class type on an action bar item.
	GetPopupClass (see page 2978)	Returns the popup class added to a menu action band.
	GetScrollBtnClass (see page 2978)	This is GetScrollBtnClass, a member of class TPlatformDefaultStyleActionBars.
	GetStyleName (see page 2978)	Returns the style associated with action manager.

55.1.1.2.1 TPlatformDefaultStyleActionBars.GetAddRemoveItemClass Method

Returns the base class for ActionBand menus for customizing action bands.

Pascal

```
function GetAddRemoveItemClass(ActionBar: TCustomActionBar): TCustomAddRemoveItemClass;
override;
```

C++

```
virtual __fastcall TCustomAddRemoveItemClass GetAddRemoveItemClass(TCustomActionBar ActionBar);
```

Description

Call GetAddRemoveItemClass to customize action bands.

See Also

TCustomAddRemoveItem

55.1.1.2.2 TPlatformDefaultStyleActionBars.GetColorMapClass Method

Returns the class type of the colormap for an action bar.

Pascal

```
function GetColorMapClass(ActionBar: TCustomActionBar): TCustomColorMapClass; override;
```

C++

```
virtual __fastcall TCustomColorMapClass GetColorMapClass(TCustomActionBar ActionBar);
```

Description

Returns the class type TCustomActionBarColorMap, which is a colormap associated with an action bar. Override this method to customize the appearance of your applications.

55.1.1.2.3 TPlatformDefaultStyleActionBars.GetControlClass Method

Returns the control class type on an action bar item.

Pascal

```
function GetControlClass(ActionBar: TCustomActionBar; AnItem: TActionClientItem):
TCustomActionControlClass; override;
```

C++

```
virtual __fastcall TCustomActionControlClass GetControlClass(TCustomActionBar ActionBar,
TActionClientItem AnItem);
```

Description

Returns the control class type on an item in the action bar, such as a button or menu item.

55.1.1.2.4 TPlatformDefaultStyleActionBars.GetPopupClass Method

Returns the popup class added to a menu action band.

Pascal

```
function GetPopupClass(ActionBar: TCustomActionBar): TCustomPopupClass; override;
```

C++

```
virtual __fastcall TCustomPopupClass GetPopupClass(TCustomActionBar ActionBar);
```

Description

Call GetPopupClass to find the popup menu associated with a menu action band. Override GetPopupClass to change which popup class is associated with a menu action band depending on the state of the control.

See Also

TCustomActionPopupMenu

55.1.1.2.5 TPlatformDefaultStyleActionBars.GetScrollBtnClass Method

Pascal

```
function GetScrollBtnClass: TCustomToolScrollBtnClass; override;
```

C++

```
virtual __fastcall TCustomToolScrollBtnClass GetScrollBtnClass();
```

Description

This is GetScrollBtnClass, a member of class TPlatformDefaultStyleActionBars.

55.1.1.2.6 TPlatformDefaultStyleActionBars.GetStyleName Method

Returns the style associated with action manager.

Pascal

```
function GetStyleName: string; override;
```

C++

```
virtual __fastcall UnicodeString GetStyleName();
```

Description

Returns the value of the Style property.

55.2 Variables

The following table lists variables in this documentation.

Variables

Name	Description
PlatformDefaultStyle (see page 2979)	This is variable PlatformDefaultStyleActnCtrls.PlatformDefaultStyle.

55.2.1 PlatformDefaultStyleActnCtrls.PlatformDefaultStyle Variable

Pascal

```
PlatformDefaultStyle: TPlatformDefaultStyleActionBars;
```

C++

```
TPlatformDefaultStyleActionBars PlatformDefaultStyle;
```

File

PlatformDefaultStyleActnCtrls

Description

This is variable PlatformDefaultStyleActnCtrls.PlatformDefaultStyle.

56 pngimage Namespace

This is namespace pngimage.

Topics

Name	Description
Classes (see page 2984)	The following table lists classes in this documentation.
Functions (see page 3009)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 3020)	The following table lists structs, records, enums in this documentation.
Types (see page 3023)	The following table lists types in this documentation.
Constants (see page 3029)	The following table lists constants in this documentation.

Classes

	Name	Description
	EPNGCannotChangeTransparent (see page 2984)	This is class pngimage.EPNGCannotChangeTransparent.
	EPNGCouldNotLoadResource (see page 2985)	This is class pngimage.EPNGCouldNotLoadResource.
	EPngError (see page 2985)	This is class pngimage.EPngError.
	EPNGHeaderNotPresent (see page 2985)	This is class pngimage.EPNGHeaderNotPresent.
	EPNGIHDRNotFirst (see page 2986)	This is class pngimage.EPNGIHDRNotFirst.
	EPngInvalidCRC (see page 2986)	This is class pngimage.EPngInvalidCRC.
	EPNGInvalidFileHeader (see page 2987)	This is class pngimage.EPNGInvalidFileHeader.
	EPngInvalidIHDR (see page 2987)	This is class pngimage.EPngInvalidIHDR.
	EPNGInvalidNewSize (see page 2987)	This is class pngimage.EPNGInvalidNewSize.
	EPNGInvalidPalette (see page 2988)	This is class pngimage.EPNGInvalidPalette.
	EPNGInvalidSpec (see page 2988)	This is class pngimage.EPNGInvalidSpec.
	EPNGMissingMultipleIDAT (see page 2988)	This is class pngimage.EPNGMissingMultipleIDAT.
	EPNGMissingPalette (see page 2989)	This is class pngimage.EPNGMissingPalette.
	EPNGNoImageData (see page 2989)	This is class pngimage.EPNGNoImageData.
	EPNGNotExists (see page 2990)	This is class pngimage.EPNGNotExists.
	EPNGOutMemory (see page 2990)	This is class pngimage.EPNGOutMemory.
	EPNGSizeExceeds (see page 2990)	This is class pngimage.EPNGSizeExceeds.
	EPngUnexpectedEnd (see page 2991)	This is class pngimage.EPngUnexpectedEnd.
	EPNGUnknownCompression (see page 2991)	This is class pngimage.EPNGUnknownCompression.
	EPNGUnknownCriticalChunk (see page 2991)	This is class pngimage.EPNGUnknownCriticalChunk.
	EPNGUnknownInterlace (see page 2992)	This is class pngimage.EPNGUnknownInterlace.
	EPNGZLIBError (see page 2992)	This is class pngimage.EPNGZLIBError.
	Exception (see page 2993)	This is class pngimage.Exception.

	TFileStream (see page 2993)	This is class pngimage.TFileStream.
	TPngImage (see page 2996)	This is class pngimage.TPngImage.
	TPNGList (see page 2997)	This is class pngimage.TPNGList.
	TPNGPointerList (see page 2999)	This is class pngimage.TPNGPointerList.
	TResourceStream (see page 3003)	This is class pngimage.TResourceStream.
	TStream (see page 3005)	This is class pngimage.TStream.

Constants

Name	Description
COLOR_GRAYSCALE (see page 3029)	This is constant pngimage.COLOR_GRAYSCALE.
COLOR_GRAYSCALEALPHA (see page 3029)	This is constant pngimage.COLOR_GRAYSCALEALPHA.
COLOR_PALETTE (see page 3030)	This is constant pngimage.COLOR_PALETTE.
COLOR_RGB (see page 3030)	This is constant pngimage.COLOR_RGB.
COLOR_RGBALPHA (see page 3030)	This is constant pngimage.COLOR_RGBALPHA.
FILTER_AVERAGE (see page 3031)	This is constant pngimage.FILTER_AVERAGE.
FILTER_NONE (see page 3031)	This is constant pngimage.FILTER_NONE.
FILTER_PAETH (see page 3031)	This is constant pngimage.FILTER_PAETH.
FILTER_SUB (see page 3032)	This is constant pngimage.FILTER_SUB.
FILTER_UP (see page 3032)	This is constant pngimage.FILTER_UP.
Index (see page 3032)	This is constant pngimage.Index.
LibraryVersion (see page 3033)	This is constant pngimage.LibraryVersion.
soFromBeginning (see page 3033)	This is constant pngimage.soFromBeginning.
soFromCurrent (see page 3033)	This is constant pngimage.soFromCurrent.
soFromEnd (see page 3034)	This is constant pngimage.soFromEnd.
Z_FINISH (see page 3034)	This is constant pngimage.Z_FINISH.
Z_NO_FLUSH (see page 3034)	This is constant pngimage.Z_NO_FLUSH.
Z_STREAM_END (see page 3035)	This is constant pngimage.Z_STREAM_END.
ZLIBErrors (see page 3035)	This is constant pngimage.ZLIBErrors.

Functions

	Name	Description
	AddtEXt (see page 3009)	This is function pngimage.AddtEXt.
	AddzTXt (see page 3010)	This is function pngimage.AddzTXt.
	Assign (see page 3010)	This is function pngimage.Assign.
	AssignHandle (see page 3010)	This is function pngimage.AssignHandle.
	AssignPNG (see page 3011)	This is function pngimage.AssignPNG.
	CreateAlpha (see page 3011)	This is function pngimage.CreateAlpha.
	DoSetPalette (see page 3011)	This is function pngimage.DoSetPalette.
	Draw (see page 3012)	This is function pngimage.Draw.
	DrawPartialTrans (see page 3012)	This is function pngimage.DrawPartialTrans.
	DrawUsingPixelInformation (see page 3012)	This is function pngimage.DrawUsingPixelInformation.
	GetAlphaScanline (see page 3013)	This is function pngimage.GetAlphaScanline.
	GetEmpty (see page 3013)	This is function pngimage.GetEmpty.
	GetLibraryVersion (see page 3013)	This is function pngimage.GetLibraryVersion.
	GetPalette (see page 3014)	This is function pngimage.GetPalette.
	GetPixelInformation (see page 3014)	This is function pngimage.GetPixelInformation.
	GetPixels (see page 3014)	This is function pngimage.GetPixels.
	GetScanline (see page 3015)	This is function pngimage.GetScanline.
	GetSupportsPartialTransparency (see page 3015)	This is function pngimage.GetSupportsPartialTransparency.
	GetTransparencyMode (see page 3015)	This is function pngimage.GetTransparencyMode.
	GetTransparentColor (see page 3016)	This is function pngimage.GetTransparentColor.
	GetWidth (see page 3016)	This is function pngimage.GetWidth.
	HasPixelInformation (see page 3016)	This is function pngimage.HasPixelInformation.
	RaiseError (see page 3017)	This is function pngimage.RaiseError.
	RemoveTransparency (see page 3017)	This is function pngimage.RemoveTransparency.

	Resize (see page 3017)	This is function pngimage.Resize.
	SetMaxIdatSize (see page 3018)	This is function pngimage.SetMaxIdatSize.
	SetPixels (see page 3018)	This is function pngimage.SetPixels.
	SetTransparentColor (see page 3018)	This is function pngimage.SetTransparentColor.
	SetWidth (see page 3019)	This is function pngimage.SetWidth.

Structs, Records, Enums

	Name	Description
	TFileStreamMode (see page 3020)	This is record pngimage.TFileStreamMode.
	TFilter (see page 3020)	This is record pngimage.TFilter.
	TInterlaceMethod (see page 3021)	This is record pngimage.TInterlaceMethod.
	TMAXBITMAPINFO (see page 3021)	This is record pngimage.TMAXBITMAPINFO.
	TPNGTransparencyMode (see page 3022)	This is record pngimage.TPNGTransparencyMode.
	TRGBPixel (see page 3022)	This is record pngimage.TRGBPixel.

Types

Name	Description
ExceptClass (see page 3023)	This is type pngimage.ExceptClass.
pByteArray (see page 3023)	This is type pngimage.pByteArray.
pCardinal (see page 3024)	This is type pngimage.pCardinal.
pPointerArray (see page 3024)	This is type pngimage.pPointerArray.
pRGBLine (see page 3024)	This is type pngimage.pRGBLine.
pRGBPixel (see page 3025)	This is type pngimage.pRGBPixel.
TBitmap (see page 3025)	This is type pngimage.TBitmap.
TByteArray (see page 3025)	This is type pngimage.TByteArray.
TCanvas (see page 3026)	This is type pngimage.TCanvas.
TChunkClass (see page 3026)	This is type pngimage.TChunkClass.
TColor (see page 3026)	This is type pngimage.TColor.
TCompressionLevel (see page 3027)	This is type pngimage.TCompressionLevel.
TFileStreamModeSet (see page 3027)	This is type pngimage.TFileStreamModeSet.
TFilters (see page 3027)	This is type pngimage.TFilters.
TPersistent (see page 3028)	This is type pngimage.TPersistent.
TPointerArray (see page 3028)	This is type pngimage.TPointerArray.
TRGBLine (see page 3028)	This is type pngimage.TRGBLine.

56.1 Classes

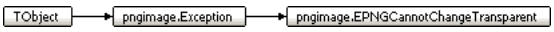
The following table lists classes in this documentation.

Classes

	Name	Description
	EPNGCannotChangeTransparent (see page 2984)	This is class pngimage.EPNGCannotChangeTransparent.
	EPNGCouldNotLoadResource (see page 2985)	This is class pngimage.EPNGCouldNotLoadResource.
	EPngError (see page 2985)	This is class pngimage.EPngError.
	EPNGHeaderNotPresent (see page 2985)	This is class pngimage.EPNGHeaderNotPresent.
	EPNGIHDRNotFirst (see page 2986)	This is class pngimage.EPNGIHDRNotFirst.
	EPngInvalidCRC (see page 2986)	This is class pngimage.EPngInvalidCRC.
	EPNGInvalidFileHeader (see page 2987)	This is class pngimage.EPNGInvalidFileHeader.
	EPngInvalidIHDR (see page 2987)	This is class pngimage.EPngInvalidIHDR.
	EPNGInvalidNewSize (see page 2987)	This is class pngimage.EPNGInvalidNewSize.
	EPNGInvalidPalette (see page 2988)	This is class pngimage.EPNGInvalidPalette.
	EPNGInvalidSpec (see page 2988)	This is class pngimage.EPNGInvalidSpec.
	EPNGMissingMultipleIDAT (see page 2988)	This is class pngimage.EPNGMissingMultipleIDAT.
	EPNGMissingPalette (see page 2989)	This is class pngimage.EPNGMissingPalette.
	EPNGNoImageData (see page 2989)	This is class pngimage.EPNGNoImageData.
	EPNGNotExists (see page 2990)	This is class pngimage.EPNGNotExists.
	EPNGOutMemory (see page 2990)	This is class pngimage.EPNGOutMemory.
	EPNGSizeExceeds (see page 2990)	This is class pngimage.EPNGSizeExceeds.
	EPngUnexpectedEnd (see page 2991)	This is class pngimage.EPngUnexpectedEnd.
	EPNGUnknownCompression (see page 2991)	This is class pngimage.EPNGUnknownCompression.
	EPNGUnknownCriticalChunk (see page 2991)	This is class pngimage.EPNGUnknownCriticalChunk.
	EPNGUnknownInterlace (see page 2992)	This is class pngimage.EPNGUnknownInterlace.
	EPNGZLIBError (see page 2992)	This is class pngimage.EPNGZLIBError.
	Exception (see page 2993)	This is class pngimage.Exception.
	TFileStream (see page 2993)	This is class pngimage.TFileStream.
	TPngImage (see page 2996)	This is class pngimage.TPngImage.
	TPNGList (see page 2997)	This is class pngimage.TPNGList.
	TPNGPointerList (see page 2999)	This is class pngimage.TPNGPointerList.
	TResourceStream (see page 3003)	This is class pngimage.TResourceStream.
	TStream (see page 3005)	This is class pngimage.TStream.

56.1.1 EPNGCannotChangeTransparent Class

Class Hierarchy



Pascal

```
EPNGCannotChangeTransparent = class(Exception);
```

C++

```
class EPNGCannotChangeTransparent : public Exception;
```

File

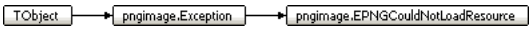
pngimage

Description

This is class pngimage.EPNGCannotChangeTransparent.

56.1.2 EPNGCouldNotLoadResource Class

Class Hierarchy



Pascal

```
EPNGCouldNotLoadResource = class(Exception);
```

C++

```
class EPNGCouldNotLoadResource : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGCouldNotLoadResource.

56.1.3 EPngError Class

Class Hierarchy

Pascal

```
EPngError = class(Exception);
```

C++

```
class EPngError : public Exception;
```

File

pngimage

Description

This is class pngimage.EPngError.

56.1.4 EPNGHeaderNotPresent Class

Class Hierarchy

Pascal

```
EPNGHeaderNotPresent = class(Exception);
```

C++

```
class EPNGHeaderNotPresent : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGHeaderNotPresent.

56.1.5 EPNGIHDRNotFirst Class

Class Hierarchy

Pascal

```
EPNGIHDRNotFirst = class(Exception);
```

C++

```
class EPNGIHDRNotFirst : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGIHDRNotFirst.

56.1.6 EPngInvalidCRC Class

Class Hierarchy

Pascal

```
EPngInvalidCRC = class(Exception);
```

C++

```
class EPngInvalidCRC : public Exception;
```

File

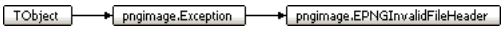
pngimage

Description

This is class pngimage.EPngInvalidCRC.

56.1.7 EPNGInvalidFileHeader Class

Class Hierarchy



Pascal

```
EPNGInvalidFileHeader = class(Exception);
```

C++

```
class EPNGInvalidFileHeader : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGInvalidFileHeader.

56.1.8 EPngInvalidIHDR Class

Class Hierarchy

Pascal

```
EPngInvalidIHDR = class(Exception);
```

C++

```
class EPngInvalidIHDR : public Exception;
```

File

pngimage

Description

This is class pngimage.EPngInvalidIHDR.

56.1.9 EPNGInvalidNewSize Class

Class Hierarchy

Pascal

```
EPNGInvalidNewSize = class(Exception);
```

C++

```
class EPNGInvalidNewSize : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGInvalidNewSize.

56.1.10 EPNGInvalidPalette Class

Class Hierarchy

Pascal

```
EPNGInvalidPalette = class(Exception);
```

C++

```
class EPNGInvalidPalette : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGInvalidPalette.

56.1.11 EPNGInvalidSpec Class

Class Hierarchy

Pascal

```
EPNGInvalidSpec = class(Exception);
```

C++

```
class EPNGInvalidSpec : public Exception;
```

File

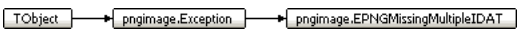
pngimage

Description

This is class pngimage.EPNGInvalidSpec.

56.1.12 EPNGMissingMultipleIDAT Class

Class Hierarchy



Pascal

```
EPNGMissingMultipleIDAT = class(Exception);
```

C++

```
class EPNGMissingMultipleIDAT : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGMissingMultipleIDAT.

56.1.13 EPNGMissingPalette Class

Class Hierarchy

Pascal

```
EPNGMissingPalette = class(Exception);
```

C++

```
class EPNGMissingPalette : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGMissingPalette.

56.1.14 EPNGNoImageData Class

Class Hierarchy

Pascal

```
EPNGNoImageData = class(Exception);
```

C++

```
class EPNGNoImageData : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGNoImageData.

56.1.15 EPNGNotExists Class

Class Hierarchy

Pascal

```
EPNGNotExists = class(Exception);
```

C++

```
class EPNGNotExists : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGNotExists.

56.1.16 EPNGOutMemory Class

Class Hierarchy

Pascal

```
EPNGOutMemory = class(Exception);
```

C++

```
class EPNGOutMemory : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGOutMemory.

56.1.17 EPNGSizeExceeds Class

Class Hierarchy

Pascal

```
EPNGSizeExceeds = class(Exception);
```

C++

```
class EPNGSizeExceeds : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGSizeExceeds.

56.1.18 EPngUnexpectedEnd Class

Class Hierarchy

Pascal

```
EPngUnexpectedEnd = class(Exception);
```

C++

```
class EPngUnexpectedEnd : public Exception;
```

File

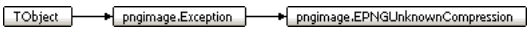
pngimage

Description

This is class pngimage.EPngUnexpectedEnd.

56.1.19 EPNGUnknownCompression Class

Class Hierarchy



Pascal

```
EPNGUnknownCompression = class(Exception);
```

C++

```
class EPNGUnknownCompression : public Exception;
```

File

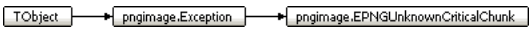
pngimage

Description

This is class pngimage.EPNGUnknownCompression.

56.1.20 EPNGUnknownCriticalChunk Class

Class Hierarchy



Pascal

```
EPNGUnknownCriticalChunk = class(Exception);
```

C++

```
class EPNGUnknownCriticalChunk : public Exception;
```

File

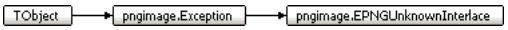
pngimage

Description

This is class pngimage.EPNGUnknownCriticalChunk.

56.1.21 EPNGUnknownInterlace Class

Class Hierarchy



Pascal

```
EPNGUnknownInterlace = class(Exception);
```

C++

```
class EPNGUnknownInterlace : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGUnknownInterlace.

56.1.22 EPNGZLIBError Class

Class Hierarchy

Pascal

```
EPNGZLIBError = class(Exception);
```

C++

```
class EPNGZLIBError : public Exception;
```

File

pngimage

Description

This is class pngimage.EPNGZLIBError.

56.1.23 Exception Class

Class Hierarchy

Pascal

```
Exception = class(TObject);
```

C++

```
class Exception : public TObject;
```

File

pngimage

Description

This is class pngimage.Exception.

56.1.23.1 Exception Members

The following tables list the members exposed by Exception.

Public Methods

	Name	Description
	Create (see page 2993)	This is Create, a member of class Exception.

56.1.23.2 Exception.Create Constructor

Pascal

```
constructor Create(Msg: String);
```

C++

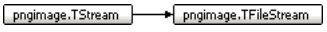
```
__fastcall Exception(UnicodeString Msg);
```

Description

This is Create, a member of class Exception.

56.1.24 TFileStream Class

Class Hierarchy



Pascal

```
TFileStream = class(TStream);
```

C++

```
class TFileStream : public TStream;
```

File

pngimage

Description

This is class pngimage.TFileStream.

56.1.24.1 TFileStream Members

The following tables list the members exposed by TFileStream.

Protected Methods

	Name	Description
	GetPosition (see page 3006)	This is GetPosition, a member of class TStream.
	GetSize (see page 3006)	This is GetSize, a member of class TStream.
	SetPosition (see page 3007)	This is SetPosition, a member of class TStream.
	SetSize (see page 2995)	This is SetSize, a member of class TFileStream.

Public Methods

	Name	Description
	CopyFrom (see page 3006)	This is CopyFrom, a member of class TStream.
	Create (see page 2994)	This is Create, a member of class TFileStream.
	Destroy (see page 2994)	This is Destroy, a member of class TFileStream.
	Read (see page 2995)	This is Read, a member of class TFileStream.
	Seek (see page 2995)	This is Seek, a member of class TFileStream.
	Write (see page 2996)	This is Write, a member of class TFileStream.

Public Properties

	Name	Description
	Position (see page 3008)	This is Position, a member of class TStream.
	Size (see page 3008)	This is Size, a member of class TStream.

56.1.24.2 TFileStream.Create Constructor

Pascal

```
constructor Create(Filename: String; Mode: TFileStreamModeSet);
```

C++

```
__fastcall TFileStream(UnicodeString Filename, TFileStreamModeSet Mode);
```

Description

This is Create, a member of class TFileStream.

56.1.24.3 TFileStream.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TFileStream();
```

Description

This is Destroy, a member of class TFileStream.

56.1.24.4 TFileStream Methods

The methods of the TFileStream class are listed here.

Protected Methods

	Name	Description
	SetSize (see page 2995)	This is SetSize, a member of class TFileStream.

Public Methods

	Name	Description
	Read (see page 2995)	This is Read, a member of class TFileStream.
	Seek (see page 2995)	This is Seek, a member of class TFileStream.
	Write (see page 2996)	This is Write, a member of class TFileStream.

56.1.24.4.1 TFileStream.Read Method

Pascal

```
function Read(var Buffer; Count: Longint): Cardinal; override;
```

C++

```
virtual __fastcall unsigned Read( Buffer, Longint Count);
```

Description

This is Read, a member of class TFileStream.

56.1.24.4.2 TFileStream.Seek Method

Pascal

```
function Seek(Offset: Longint; Origin: Word): Longint; override;
```

C++

```
virtual __fastcall Longint Seek(Longint Offset, Word Origin);
```

Description

This is Seek, a member of class TFileStream.

56.1.24.4.3 TFileStream.SetSize Method

Pascal

```
procedure SetSize(const Value: Longint); override;
```

C++

```
virtual __fastcall SetSize(const Longint Value);
```

Description

This is SetSize, a member of class TFileStream.

56.1.24.4.4 TFileStream.Write Method

Pascal

```
function Write(const Buffer; Count: Longint): Cardinal; override;
```

C++

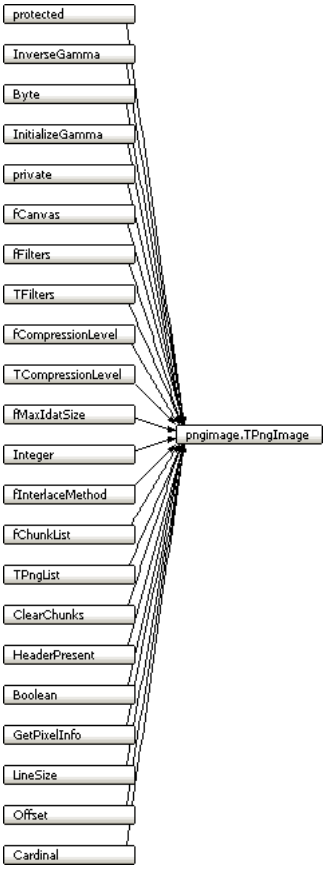
```
virtual __fastcall unsigned Write(const  Buffer, Longint Count);
```

Description

This is Write, a member of class TFileStream.

56.1.25 TPngImage Class

Class Hierarchy



Pascal

```
TPngImage = class(protected, InverseGamma, Byte, InitializeGamma, private, fCanvas, fFilters,
TFilters, fCompressionLevel, TCompressionLevel, fMaxIdatSize, Integer, fInterlaceMethod,
TInterlaceMethod, fChunkList, TPngList, ClearChunks, HeaderPresent, Boolean, GetPixelInfo,
```

```
LineSize, Offset, Cardinal);
```

C++

```
class TPngImage : public protected, public InverseGamma, public Byte, public InitializeGamma,
public private, public fCanvas, public fFilters, public TFilters, public fCompressionLevel,
public TCompressionLevel, public fMaxIdatSize, public Integer, public fInterlaceMethod, public
TInterlaceMethod, public fChunkList, public TPngList, public ClearChunks, public
HeaderPresent, public Boolean, public GetPixelInfo, public LineSize, public Offset, public
Cardinal;
```

File

pngimage

Description

This is class pngimage.TPngImage.

56.1.26 TPNGList Class

Class Hierarchy

Pascal

```
TPNGList = class(TPNGPointerList);
```

C++

```
class TPNGList : public TPNGPointerList;
```

File

pngimage

Description

This is class pngimage.TPNGList.

56.1.26.1 TPNGList Members

The following tables list the members exposed by TPNGList.

Protected Methods

	Name	Description
	Insert (see page 3001)	This is Insert, a member of class TPNGPointerList.
V	Remove (see page 3001)	This is Remove, a member of class TPNGPointerList.
	SetSize (see page 3001)	This is SetSize, a member of class TPNGPointerList.

Protected Properties

	Name	Description
R	Owner (see page 3002)	This is Owner, a member of class TPNGPointerList.

Public Methods

	Name	Description
	Add (see page 2998)	This is Add, a member of class TPNGList.
	Create (see page 3000)	This is Create, a member of class TPNGPointerList.
V	Destroy (see page 3000)	This is Destroy, a member of class TPNGPointerList.

	FindChunk (see page 2998)	This is FindChunk, a member of class TPNGList.
	ItemFromClass (see page 2998)	This is ItemFromClass, a member of class TPNGList.
	RemoveChunk (see page 2999)	This is RemoveChunk, a member of class TPNGList.

Public Properties

	Name	Description
	Count (see page 3002)	This is Count, a member of class TPNGPointerList.
R	Item (see page 2999)	This is Item, a member of class TPNGList.

56.1.26.2 TPNGList Methods

The methods of the TPNGList class are listed here.

Public Methods

	Name	Description
	Add (see page 2998)	This is Add, a member of class TPNGList.
	FindChunk (see page 2998)	This is FindChunk, a member of class TPNGList.
	ItemFromClass (see page 2998)	This is ItemFromClass, a member of class TPNGList.
	RemoveChunk (see page 2999)	This is RemoveChunk, a member of class TPNGList.

56.1.26.2.1 TPNGList.Add Method

Pascal

```
function Add(ChunkClass: TChunkClass): TChunk;
```

C++

```
__fastcall TChunk Add(TChunkClass ChunkClass);
```

Description

This is Add, a member of class TPNGList.

56.1.26.2.2 TPNGList.FindChunk Method

Pascal

```
function FindChunk(ChunkClass: TChunkClass): TChunk;
```

C++

```
__fastcall TChunk FindChunk(TChunkClass ChunkClass);
```

Description

This is FindChunk, a member of class TPNGList.

56.1.26.2.3 TPNGList.ItemFromClass Method

Pascal

```
function ItemFromClass(ChunkClass: TChunkClass): TChunk;
```

C++

```
__fastcall TChunk ItemFromClass(TChunkClass ChunkClass);
```

Description

This is ItemFromClass, a member of class TPNGList.

56.1.26.2.4 TPNGList.RemoveChunk Method

Pascal

```
procedure RemoveChunk(Chunk: TChunk); overload;
```

C++

```
__fastcall RemoveChunk(TChunk Chunk);
```

Description

This is RemoveChunk, a member of class TPNGList.

56.1.26.3 TPNGList Properties

The properties of the TPNGList class are listed here.

Public Properties

	Name	Description
	Item (see page 2999)	This is Item, a member of class TPNGList.

56.1.26.3.1 TPNGList.Item Property

Pascal

```
property Item [Index: Cardinal]: TChunk;
```

C++

```
__property TChunk Item[unsigned Index];
```

Description

This is Item, a member of class TPNGList.

56.1.27 TPNGPointerList Class

Class Hierarchy

Pascal

```
TPNGPointerList = class;
```

C++

```
class TPNGPointerList;
```

File

pngimage

Description

This is class pngimage.TPNGPointerList.

56.1.27.1 TPNGPointerList Members

The following tables list the members exposed by TPNGPointerList.

Protected Methods

	Name	Description
	Add (see page 3001)	This is Add, a member of class TPNGPointerList.
	Insert (see page 3001)	This is Insert, a member of class TPNGPointerList.
V	Remove (see page 3001)	This is Remove, a member of class TPNGPointerList.
	SetSize (see page 3001)	This is SetSize, a member of class TPNGPointerList.

Protected Properties

	Name	Description
	Item (see page 3002)	This is Item, a member of class TPNGPointerList.
R	Owner (see page 3002)	This is Owner, a member of class TPNGPointerList.

Public Methods

	Name	Description
	Create (see page 3000)	This is Create, a member of class TPNGPointerList.
V	Destroy (see page 3000)	This is Destroy, a member of class TPNGPointerList.

Public Properties

	Name	Description
	Count (see page 3002)	This is Count, a member of class TPNGPointerList.

56.1.27.2 TPNGPointerList.Create Constructor

Pascal

```
constructor Create(AOwner: TPngImage);
```

C++

```
__fastcall TPNGPointerList(TPngImage AOwner);
```

Description

This is Create, a member of class TPNGPointerList.

56.1.27.3 TPNGPointerList.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TPNGPointerList();
```

Description

This is Destroy, a member of class TPNGPointerList.

56.1.27.4 TPNGPointerList Methods

The methods of the TPNGPointerList class are listed here.

Protected Methods

	Name	Description
	Add (see page 3001)	This is Add, a member of class TPNGPointerList.
	Insert (see page 3001)	This is Insert, a member of class TPNGPointerList.
	Remove (see page 3001)	This is Remove, a member of class TPNGPointerList.
	SetSize (see page 3001)	This is SetSize, a member of class TPNGPointerList.

56.1.27.4.1 TPNGPointerList.Add Method

Pascal

```
procedure Add(Value: Pointer);
```

C++

```
__fastcall Add(void * Value);
```

Description

This is Add, a member of class TPNGPointerList.

56.1.27.4.2 TPNGPointerList.Insert Method

Pascal

```
procedure Insert(Value: Pointer; Position: Cardinal);
```

C++

```
__fastcall Insert(void * Value, unsigned Position);
```

Description

This is Insert, a member of class TPNGPointerList.

56.1.27.4.3 TPNGPointerList.Remove Method

Pascal

```
function Remove(Value: Pointer): Pointer; virtual;
```

C++

```
virtual __fastcall void * Remove(void * Value);
```

Description

This is Remove, a member of class TPNGPointerList.

56.1.27.4.4 TPNGPointerList.SetSize Method

Pascal

```
procedure SetSize(const Size: Cardinal);
```

C++

```
__fastcall SetSize(const unsigned Size);
```

Description

This is SetSize, a member of class TPNGPointerList.

56.1.27.5 TPNGPointerList Properties

The properties of the TPNGPointerList class are listed here.

Protected Properties

	Name	Description
	Item (see page 3002)	This is Item, a member of class TPNGPointerList.
R	Owner (see page 3002)	This is Owner, a member of class TPNGPointerList.

Public Properties

	Name	Description
	Count (see page 3002)	This is Count, a member of class TPNGPointerList.

56.1.27.5.1 TPNGPointerList.Count Property

Pascal

```
property Count: Cardinal;
```

C++

```
__property unsigned Count;
```

Description

This is Count, a member of class TPNGPointerList.

56.1.27.5.2 TPNGPointerList.Item Property

Pascal

```
property Item [Index: Cardinal]: Pointer;
```

C++

```
__property void * Item[unsigned Index];
```

Description

This is Item, a member of class TPNGPointerList.

56.1.27.5.3 TPNGPointerList.Owner Property

Pascal

```
property Owner: TPngImage;
```

C++

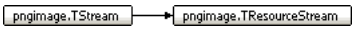
```
__property TPngImage Owner;
```

Description

This is Owner, a member of class TPNGPointerList.

56.1.28 TResourceStream Class

Class Hierarchy



Pascal

```
TResourceStream = class(TStream);
```

C++

```
class TResourceStream : public TStream;
```

File

pngimage

Description

This is class pngimage.TResourceStream.

56.1.28.1 TResourceStream Members

The following tables list the members exposed by TResourceStream.

Protected Methods

	Name	Description
	GetPosition (see page 3006)	This is GetPosition, a member of class TStream.
	GetSize (see page 3006)	This is GetSize, a member of class TStream.
	SetPosition (see page 3007)	This is SetPosition, a member of class TStream.
	SetSize (see page 3004)	This is SetSize, a member of class TResourceStream.

Public Methods

	Name	Description
	CopyFrom (see page 3006)	This is CopyFrom, a member of class TStream.
	Create (see page 3003)	This is Create, a member of class TResourceStream.
	Read (see page 3004)	This is Read, a member of class TResourceStream.
	Seek (see page 3004)	This is Seek, a member of class TResourceStream.
	Write (see page 3005)	This is Write, a member of class TResourceStream.

Public Properties

	Name	Description
	Position (see page 3008)	This is Position, a member of class TStream.
	Size (see page 3008)	This is Size, a member of class TStream.

56.1.28.2 TResourceStream.Create Constructor

Pascal

```
constructor Create(Instance: HInst; const ResName: String; ResType: PChar);
```

C++

```
__fastcall TResourceStream(HInst Instance, const UnicodeString ResName, const char * ResType);
```

Description

This is Create, a member of class TResourceStream.

56.1.28.3 TResourceStream Methods

The methods of the TResourceStream class are listed here.

Protected Methods

	Name	Description
	SetSize (see page 3004)	This is SetSize, a member of class TResourceStream.

Public Methods

	Name	Description
	Read (see page 3004)	This is Read, a member of class TResourceStream.
	Seek (see page 3004)	This is Seek, a member of class TResourceStream.
	Write (see page 3005)	This is Write, a member of class TResourceStream.

56.1.28.3.1 TResourceStream.Read Method

Pascal

```
function Read(var Buffer; Count: Integer): Cardinal; override;
```

C++

```
virtual __fastcall unsigned Read( Buffer, int Count);
```

Description

This is Read, a member of class TResourceStream.

56.1.28.3.2 TResourceStream.Seek Method

Pascal

```
function Seek(Offset: Integer; Origin: Word): Longint; override;
```

C++

```
virtual __fastcall Longint Seek(int Offset, Word Origin);
```

Description

This is Seek, a member of class TResourceStream.

56.1.28.3.3 TResourceStream.SetSize Method

Pascal

```
procedure SetSize(const Value: Longint); override;
```

C++

```
virtual __fastcall SetSize(const Longint Value);
```

Description

This is SetSize, a member of class TResourceStream.

56.1.28.3.4 TResourceStream.Write Method

Pascal

```
function Write(const Buffer; Count: Longint): Cardinal; override;
```

C++

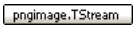
```
virtual __fastcall unsigned Write(const  Buffer, Longint Count);
```

Description

This is Write, a member of class TResourceStream.

56.1.29 TStream Class

Class Hierarchy



Pascal

```
TStream = class;
```

C++

```
class TStream;
```

File

pngimage

Description

This is class pngimage.TStream.

56.1.29.1 TStream Members

The following tables list the members exposed by TStream.

Protected Methods

	Name	Description
	GetPosition (see page 3006)	This is GetPosition, a member of class TStream.
	GetSize (see page 3006)	This is GetSize, a member of class TStream.
	SetPosition (see page 3007)	This is SetPosition, a member of class TStream.
	SetSize (see page 3007)	This is SetSize, a member of class TStream.

Public Methods

	Name	Description
	CopyFrom (see page 3006)	This is CopyFrom, a member of class TStream.
	Read (see page 3007)	This is Read, a member of class TStream.
	Seek (see page 3007)	This is Seek, a member of class TStream.
	Write (see page 3007)	This is Write, a member of class TStream.

Public Properties

	Name	Description
	Position (see page 3008)	This is Position, a member of class TStream.

	Size (see page 3008)	This is Size, a member of class TStream.

56.1.29.2 TStream Methods

The methods of the TStream class are listed here.

Protected Methods

	Name	Description
	GetPosition (see page 3006)	This is GetPosition, a member of class TStream.
	GetSize (see page 3006)	This is GetSize, a member of class TStream.
	SetPosition (see page 3007)	This is SetPosition, a member of class TStream.
	SetSize (see page 3007)	This is SetSize, a member of class TStream.

Public Methods

	Name	Description
	CopyFrom (see page 3006)	This is CopyFrom, a member of class TStream.
	Read (see page 3007)	This is Read, a member of class TStream.
	Seek (see page 3007)	This is Seek, a member of class TStream.
	Write (see page 3007)	This is Write, a member of class TStream.

56.1.29.2.1 TStream.CopyFrom Method

Pascal

```
function CopyFrom(Source: TStream; Count: Cardinal): Cardinal; virtual;
```

C++

```
virtual __fastcall unsigned CopyFrom(TStream Source, unsigned Count);
```

Description

This is CopyFrom, a member of class TStream.

56.1.29.2.2 TStream.GetPosition Method

Pascal

```
function GetPosition: Longint; virtual;
```

C++

```
virtual __fastcall Longint GetPosition();
```

Description

This is GetPosition, a member of class TStream.

56.1.29.2.3 TStream.GetSize Method

Pascal

```
function GetSize: Longint; virtual;
```

C++

```
virtual __fastcall Longint GetSize();
```

Description

This is GetSize, a member of class TStream.

56.1.29.2.4 TStream.Read Method

Pascal

```
function Read(var Buffer; Count: Longint): Cardinal; virtual; abstract;
```

C++

```
virtual __fastcall unsigned Read( Buffer, Longint Count) = 0;
```

Description

This is Read, a member of class TStream.

56.1.29.2.5 TStream.Seek Method

Pascal

```
function Seek(Offset: Longint; Origin: Word): Longint; virtual; abstract;
```

C++

```
virtual __fastcall Longint Seek(Longint Offset, Word Origin) = 0;
```

Description

This is Seek, a member of class TStream.

56.1.29.2.6 TStream.SetPosition Method

Pascal

```
procedure SetPosition(const Value: Longint); virtual;
```

C++

```
virtual __fastcall SetPosition(const Longint Value);
```

Description

This is SetPosition, a member of class TStream.

56.1.29.2.7 TStream.SetSize Method

Pascal

```
procedure SetSize(const Value: Longint); virtual; abstract;
```

C++

```
virtual __fastcall SetSize(const Longint Value) = 0;
```

Description

This is SetSize, a member of class TStream.

56.1.29.2.8 TStream.Write Method

Pascal

```
function Write(const Buffer; Count: Longint): Cardinal; virtual; abstract;
```

C++

```
virtual __fastcall unsigned Write(const  Buffer, Longint Count) = 0;
```

Description

This is Write, a member of class TStream.

56.1.29.3 TStream Properties

The properties of the TStream class are listed here.

Public Properties

	Name	Description
	Position (see page 3008)	This is Position, a member of class TStream.
	Size (see page 3008)	This is Size, a member of class TStream.

56.1.29.3.1 TStream.Position Property

Pascal

```
property Position: Longint;
```

C++

```
__property Longint Position;
```

Description

This is Position, a member of class TStream.

56.1.29.3.2 TStream.Size Property

Pascal

```
property Size: Longint;
```

C++

```
__property Longint Size;
```

Description

This is Size, a member of class TStream.

56.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	AddtEXt (see page 3009)	This is function pngimage.AddtEXt.
	AddzTXt (see page 3010)	This is function pngimage.AddzTXt.
	Assign (see page 3010)	This is function pngimage.Assign.
	AssignHandle (see page 3010)	This is function pngimage.AssignHandle.
	AssignPNG (see page 3011)	This is function pngimage.AssignPNG.
	CreateAlpha (see page 3011)	This is function pngimage.CreateAlpha.
	DoSetPalette (see page 3011)	This is function pngimage.DoSetPalette.
	Draw (see page 3012)	This is function pngimage.Draw.
	DrawPartialTrans (see page 3012)	This is function pngimage.DrawPartialTrans.
	DrawUsingPixelInformation (see page 3012)	This is function pngimage.DrawUsingPixelInformation.
	GetAlphaScanline (see page 3013)	This is function pngimage.GetAlphaScanline.
	GetEmpty (see page 3013)	This is function pngimage.GetEmpty.
	GetLibraryVersion (see page 3013)	This is function pngimage.GetLibraryVersion.
	GetPalette (see page 3014)	This is function pngimage.GetPalette.
	GetPixelInformation (see page 3014)	This is function pngimage.GetPixelInformation.
	GetPixels (see page 3014)	This is function pngimage.GetPixels.
	GetScanline (see page 3015)	This is function pngimage.GetScanline.
	GetSupportsPartialTransparency (see page 3015)	This is function pngimage.GetSupportsPartialTransparency.
	GetTransparencyMode (see page 3015)	This is function pngimage.GetTransparencyMode.
	GetTransparentColor (see page 3016)	This is function pngimage.GetTransparentColor.
	GetWidth (see page 3016)	This is function pngimage.GetWidth.
	HasPixelInformation (see page 3016)	This is function pngimage.HasPixelInformation.
	RaiseError (see page 3017)	This is function pngimage.RaiseError.
	RemoveTransparency (see page 3017)	This is function pngimage.RemoveTransparency.
	Resize (see page 3017)	This is function pngimage.Resize.
	SetMaxIdatSize (see page 3018)	This is function pngimage.SetMaxIdatSize.
	SetPixels (see page 3018)	This is function pngimage.SetPixels.
	SetTransparentColor (see page 3018)	This is function pngimage.SetTransparentColor.
	SetWidth (see page 3019)	This is function pngimage.SetWidth.

56.2.1 pngimage.AddtEXt Function

Pascal

```
procedure AddtEXt(const Keyword: AnsiString; const Text: AnsiString);
```

C++

```
AddtEXt(const AnsiString Keyword, const AnsiString Text);
```

File

pngimage

Description

This is function pngimage.AddtEXt.

56.2.2 pngimage.AddzTXt Function

Pascal

```
procedure AddzTXt(const Keyword: AnsiString; const Text: AnsiString);
```

C++

```
AddzTXt(const AnsiString Keyword, const AnsiString Text);
```

File

pngimage

Description

This is function pngimage.AddzTXt.

56.2.3 pngimage.Assign Function

Pascal

```
procedure Assign(Source: TPersistent); override;
```

C++

```
Assign(TPersistent * Source);
```

File

pngimage

Description

This is function pngimage.Assign.

56.2.4 pngimage.AssignHandle Function

Pascal

```
procedure AssignHandle(Handle: HBitmap; Transparent: Boolean; TransparentColor: ColorRef);
```

C++

```
AssignHandle(HBitmap Handle, Boolean Transparent, ColorRef TransparentColor);
```

File

pngimage

Description

This is function pngimage.AssignHandle.

56.2.5 pngimage.AssignPNG Function

Pascal

```
procedure AssignPNG(Source: TPngImage);
```

C++

```
AssignPNG(TPngImage Source);
```

File

pngimage

Description

This is function pngimage.AssignPNG.

56.2.6 pngimage.CreateAlpha Function

Pascal

```
procedure CreateAlpha;
```

C++

```
CreateAlpha();
```

File

pngimage

Description

This is function pngimage.CreateAlpha.

56.2.7 pngimage.DoSetPalette Function

Pascal

```
procedure DoSetPalette(Value: HPALETTE; const UpdateColors: Boolean);
```

C++

```
DoSetPalette(HPALETTE Value, const Boolean UpdateColors);
```

File

pngimage

Description

This is function pngimage.DoSetPalette.

56.2.8 pngimage.Draw Function

Pascal

```
procedure Draw(ACanvas: TCanvas; const Rect: TRect); override;
```

C++

```
Draw(TCanvas ACanvas, const TRect Rect);
```

File

pngimage

Description

This is function pngimage.Draw.

56.2.9 pngimage.DrawPartialTrans Function

Pascal

```
procedure DrawPartialTrans(DC: HDC; Rect: TRect);
```

C++

```
DrawPartialTrans(HDC DC, TRect Rect);
```

File

pngimage

Description

This is function pngimage.DrawPartialTrans.

56.2.10 pngimage.DrawUsingPixelInformation Function

Pascal

```
procedure DrawUsingPixelInformation(Canvas: TCanvas; Point: TPoint);
```

C++

```
DrawUsingPixelInformation(TCanvas Canvas, TPoint Point);
```

File

pngimage

Description

This is function pngimage.DrawUsingPixelInformation.

56.2.11 pngimage.GetAlphaScanline Function

Pascal

```
function GetAlphaScanline(const LineIndex: Integer): pByteArray;
```

C++

```
pByteArray GetAlphaScanline(const int LineIndex);
```

File

pngimage

Description

This is function pngimage.GetAlphaScanline.

56.2.12 pngimage.GetEmpty Function

Pascal

```
function GetEmpty: Boolean; override;
```

C++

```
Boolean GetEmpty();
```

File

pngimage

Description

This is function pngimage.GetEmpty.

56.2.13 pngimage.GetLibraryVersion Function

Pascal

```
function GetLibraryVersion: String;
```

C++

```
UnicodeString GetLibraryVersion();
```

File

pngimage

Description

This is function pngimage.GetLibraryVersion.

56.2.14 pngimage.GetPalette Function

Pascal

```
function GetPalette: HPALETTE; override;
```

C++

```
HPALETTE GetPalette();
```

File

pngimage

Description

This is function pngimage.GetPalette.

56.2.15 pngimage.GetPixelInformation Function

Pascal

```
function GetPixelInformation: TChunkpHYs;
```

C++

```
TChunkpHYs GetPixelInformation();
```

File

pngimage

Description

This is function pngimage.GetPixelInformation.

56.2.16 pngimage.GetPixels Function

Pascal

```
function GetPixels(const X: Integer; const Y: Integer): TColor; virtual;
```

C++

```
virtual TColor GetPixels(const int X, const int Y);
```

File

pngimage

Description

This is function pngimage.GetPixels.

56.2.17 pngimage.GetScanline Function

Pascal

```
function GetScanline(const LineIndex: Integer): Pointer;
```

C++

```
void * GetScanline(const int LineIndex);
```

File

pngimage

Description

This is function pngimage.GetScanline.

56.2.18 pngimage.GetSupportsPartialTransparency Function

Pascal

```
function GetSupportsPartialTransparency: Boolean; override;
```

C++

```
Boolean GetSupportsPartialTransparency();
```

File

pngimage

Description

This is function pngimage.GetSupportsPartialTransparency.

56.2.19 pngimage.GetTransparencyMode Function

Pascal

```
function GetTransparencyMode: TPNGTransparencyMode;
```

C++

```
TPNGTransparencyMode GetTransparencyMode();
```

File

pngimage

Description

This is function pngimage.GetTransparencyMode.

56.2.20 pngimage.GetTransparentColor Function

Pascal

```
function GetTransparentColor: TColor;
```

C++

```
TColor GetTransparentColor();
```

File

pngimage

Description

This is function pngimage.GetTransparentColor.

56.2.21 pngimage.GetWidth Function

Pascal

```
function GetWidth: Integer; override;
```

C++

```
int GetWidth();
```

File

pngimage

Description

This is function pngimage.GetWidth.

56.2.22 pngimage.HasPixelInformation Function

Pascal

```
function HasPixelInformation: Boolean;
```

C++

```
Boolean HasPixelInformation();
```

File

pngimage

Description

This is function pngimage.HasPixelInformation.

56.2.23 pngimage.RaiseError Function

Pascal

```
procedure RaiseError(ExceptionClass: ExceptClass; Text: String);
```

C++

```
RaiseError(ExceptClass ExceptionClass, UnicodeString Text);
```

File

pngimage

Description

This is function pngimage.RaiseError.

56.2.24 pngimage.RemoveTransparency Function

Pascal

```
procedure RemoveTransparency;
```

C++

```
RemoveTransparency();
```

File

pngimage

Description

This is function pngimage.RemoveTransparency.

56.2.25 pngimage.Resize Function

Pascal

```
procedure Resize(const CX: Integer; const CY: Integer);
```

C++

```
Resize(const int CX, const int CY);
```

File

pngimage

Description

This is function pngimage.Resize.

56.2.26 pngimage.SetMaxIdatSize Function

Pascal

```
procedure SetMaxIdatSize(const Value: Integer);
```

C++

```
SetMaxIdatSize(const int Value);
```

File

pngimage

Description

This is function pngimage.SetMaxIdatSize.

56.2.27 pngimage.SetPixels Function

Pascal

```
procedure SetPixels(const X: Integer; const Y: Integer; const Value: TColor); virtual;
```

C++

```
virtual SetPixels(const int X, const int Y, const TColor Value);
```

File

pngimage

Description

This is function pngimage.SetPixels.

56.2.28 pngimage.SetTransparentColor Function

Pascal

```
procedure SetTransparentColor(const Value: TColor);
```

C++

```
SetTransparentColor(const TColor Value);
```

File

pngimage

Description

This is function pngimage.SetTransparentColor.

56.2.29 pngimage.SetWidth Function

Pascal

```
procedure SetWidth(Value: Integer); override;
```

C++

```
SetWidth(int Value);
```

File

pngimage

Description

This is function pngimage.SetWidth.

56.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TFileStreamMode (see page 3020)	This is record pngimage.TFileStreamMode.
	TFilter (see page 3020)	This is record pngimage.TFilter.
	TInterlaceMethod (see page 3021)	This is record pngimage.TInterlaceMethod.
	TPNGTransparencyMode (see page 3022)	This is record pngimage.TPNGTransparencyMode.

Records

	Name	Description
	TMAXBITMAPINFO (see page 3021)	This is record pngimage.TMAXBITMAPINFO.
	TRGBPixel (see page 3022)	This is record pngimage.TRGBPixel.

56.3.1 pngimage.TFileStreamMode Enumeration

Pascal

```
TFileStreamMode = (
  fsmRead,
  fsmWrite,
  fsmCreate
);
```

C++

```
enum TFileStreamMode {
  fsmRead,
  fsmWrite,
  fsmCreate
};
```

File

pngimage

Description

This is record pngimage.TFileStreamMode.

56.3.2 pngimage.TFilter Enumeration

Pascal

```
TFilter = (
  pfNone,
  pfSub,
  pfUp,
  pfAverage,
  pfPaeth
```

```
);
```

C++

```
enum TFilter {
  pfNone,
  pfSub,
  pfUp,
  pfAverage,
  pfPaeth
};
```

File

pngimage

Description

This is record pngimage.TFilter.

56.3.3 pngimage.TInterlaceMethod Enumeration

Pascal

```
TInterlaceMethod = (
  imNone,
  imAdam7
);
```

C++

```
enum TInterlaceMethod {
  imNone,
  imAdam7
};
```

File

pngimage

Description

This is record pngimage.TInterlaceMethod.

56.3.4 pngimage.TMAXBITMAPINFO Record

Pascal

```
TMAXBITMAPINFO = packed record
  bmiHeader: TBitmapInfoHeader;
  bmiColors: packed array[0..255] of TRGBQuad;
end;
```

C++

```
struct TMAXBITMAPINFO {
  TBitmapInfoHeader bmiHeader;
  packed array[0..255] of TRGBQuad bmiColors;
};
```

File

pngimage

Description

This is record pngimage.TMAXBITMAPINFO.

56.3.5 pngimage.TPNGTransparencyMode Enumeration

Pascal

```
TPNGTransparencyMode = (
  ptmNone,
  ptmBit,
  ptmPartial
);
```

C++

```
enum TPNGTransparencyMode {
  ptmNone,
  ptmBit,
  ptmPartial
};
```

File

pngimage

Description

This is record pngimage.TPNGTransparencyMode.

56.3.6 pngimage.TRGBPixel Record

Pascal

```
TRGBPixel = packed record
  B: Byte;
  G: Byte;
  R: Byte;
end;
```

C++

```
struct TRGBPixel {
  Byte B;
  Byte G;
  Byte R;
};
```

File

pngimage

Description

This is record pngimage.TRGBPixel.

56.4 Types

The following table lists types in this documentation.

Types

Name	Description
ExceptClass (see page 3023)	This is type pngimage.ExceptClass.
pByteArray (see page 3023)	This is type pngimage.pByteArray.
pCardinal (see page 3024)	This is type pngimage.pCardinal.
pPointerArray (see page 3024)	This is type pngimage.pPointerArray.
pRGBLine (see page 3024)	This is type pngimage.pRGBLine.
pRGBPixel (see page 3025)	This is type pngimage.pRGBPixel.
TBitmap (see page 3025)	This is type pngimage.TBitmap.
TByteArray (see page 3025)	This is type pngimage.TByteArray.
TCanvas (see page 3026)	This is type pngimage.TCanvas.
TChunkClass (see page 3026)	This is type pngimage.TChunkClass.
TColor (see page 3026)	This is type pngimage.TColor.
TCompressionLevel (see page 3027)	This is type pngimage.TCompressionLevel.
TFileStreamModeSet (see page 3027)	This is type pngimage.TFileStreamModeSet.
TFilters (see page 3027)	This is type pngimage.TFilters.
TPersistent (see page 3028)	This is type pngimage.TPersistent.
TPointerArray (see page 3028)	This is type pngimage.TPointerArray.
TRGBLine (see page 3028)	This is type pngimage.TRGBLine.

56.4.1 pngimage.ExceptClass Type

Pascal

```
ExceptClass = class of Exception;
```

C++

```
Exception ExceptClass;
```

File

pngimage

Description

This is type pngimage.ExceptClass.

56.4.2 pngimage.pByteArray Type

Pascal

```
pByteArray = ^TByteArray;
```

C++

```
^TByteArray pByteArray;
```

File

pngimage

Description

This is type pngimage.pByteArray.

56.4.3 pngimage.pCardinal Type

Pascal

```
pCardinal = ^Cardinal;
```

C++

```
^Cardinal pCardinal;
```

File

pngimage

Description

This is type pngimage.pCardinal.

56.4.4 pngimage.pPointerArray Type

Pascal

```
pPointerArray = ^TPointerArray;
```

C++

```
^TPointerArray pPointerArray;
```

File

pngimage

Description

This is type pngimage.pPointerArray.

56.4.5 pngimage.pRGBLine Type

Pascal

```
pRGBLine = ^TRGBLine;
```

C++

```
^TRGBLine pRGBLine;
```

File

pngimage

Description

This is type pngimage.pRGBLine.

56.4.6 pngimage.pRGBPixel Type

Pascal

```
pRGBPixel = ^TRGBPixel;
```

C++

```
^TRGBPixel pRGBPixel;
```

File

pngimage

Description

This is type pngimage.pRGBPixel.

56.4.7 pngimage.TBitmap Type

Pascal

```
TBitmap = HBitmap;
```

C++

```
HBitmap TBitmap;
```

File

pngimage

Description

This is type pngimage.TBitmap.

56.4.8 pngimage.TByteArray Type

Pascal

```
TByteArray = Array[Word] of Byte;
```

C++

```
Array[Word] of Byte TByteArray;
```

File

pngimage

Description

This is type pngimage.TByteArray.

56.4.9 pngimage.TCanvas Type

Pascal

```
TCanvas = THandle;
```

C++

```
THandle TCanvas;
```

File

pngimage

Description

This is type pngimage.TCanvas.

56.4.10 pngimage.TChunkClass Type

Pascal

```
TChunkClass = class of TChunk;
```

C++

```
TChunk TChunkClass;
```

File

pngimage

Description

This is type pngimage.TChunkClass.

56.4.11 pngimage.TColor Type

Pascal

```
TColor = ColorRef;
```

C++

```
ColorRef TColor;
```

File

pngimage

Description

This is type pngimage.TColor.

56.4.12 pngimage.TCompressionLevel Type

Pascal

```
TCompressionLevel = 0..9;
```

C++

```
0..9 TCompressionLevel;
```

File

pngimage

Description

This is type pngimage.TCompressionLevel.

56.4.13 pngimage.TFileStreamModeSet Type

Pascal

```
TFileStreamModeSet = set of TFileStreamMode;
```

C++

```
TFileStreamMode TFileStreamModeSet;
```

File

pngimage

Description

This is type pngimage.TFileStreamModeSet.

56.4.14 pngimage.TFilters Type

Pascal

```
TFilters = set of TFilter;
```

C++

```
TFilter TFilters;
```

File

pngimage

Description

This is type pngimage.TFilters.

56.4.15 pngimage.TPersistent Type

Pascal

```
TPersistent = TObject;
```

C++

```
TObject * TPersistent;
```

File

pngimage

Description

This is type pngimage.TPersistent.

56.4.16 pngimage.TPointerArray Type

Pascal

```
TPointerArray = Array[Word] of Pointer;
```

C++

```
Array[Word] of Pointer TPointerArray;
```

File

pngimage

Description

This is type pngimage.TPointerArray.

56.4.17 pngimage.TRGBLine Type

Pascal

```
TRGBLine = array[word] of TRGBTriple;
```

C++

```
array[word] of TRGBTriple TRGBLine;
```

File

pngimage

Description

This is type pngimage.TRGBLine.

56.5 Constants

The following table lists constants in this documentation.

Constants

Name	Description
COLOR_GRAYSCALE (see page 3029)	This is constant pngimage.COLOR_GRAYSCALE.
COLOR_GRAYSCALEALPHA (see page 3029)	This is constant pngimage.COLOR_GRAYSCALEALPHA.
COLOR_PALETTE (see page 3030)	This is constant pngimage.COLOR_PALETTE.
COLOR_RGB (see page 3030)	This is constant pngimage.COLOR_RGB.
COLOR_RGBALPHA (see page 3030)	This is constant pngimage.COLOR_RGBALPHA.
FILTER_AVERAGE (see page 3031)	This is constant pngimage.FILTER_AVERAGE.
FILTER_NONE (see page 3031)	This is constant pngimage.FILTER_NONE.
FILTER_PAETH (see page 3031)	This is constant pngimage.FILTER_PAETH.
FILTER_SUB (see page 3032)	This is constant pngimage.FILTER_SUB.
FILTER_UP (see page 3032)	This is constant pngimage.FILTER_UP.
Index (see page 3032)	This is constant pngimage.Index.
LibraryVersion (see page 3033)	This is constant pngimage.LibraryVersion.
soFromBeginning (see page 3033)	This is constant pngimage.soFromBeginning.
soFromCurrent (see page 3033)	This is constant pngimage.soFromCurrent.
soFromEnd (see page 3034)	This is constant pngimage.soFromEnd.
Z_FINISH (see page 3034)	This is constant pngimage.Z_FINISH.
Z_NO_FLUSH (see page 3034)	This is constant pngimage.Z_NO_FLUSH.
Z_STREAM_END (see page 3035)	This is constant pngimage.Z_STREAM_END.
ZLIBErrors (see page 3035)	This is constant pngimage.ZLIBErrors.

56.5.1 pngimage.COLOR_GRAYSCALE Constant

Pascal

```
COLOR_GRAYSCALE = 0;
```

C++

```
COLOR_GRAYSCALE = 0;
```

File

pngimage

Description

This is constant pngimage.COLOR_GRAYSCALE.

56.5.2 pngimage.COLOR_GRAYSCALEALPHA Constant

Pascal

```
COLOR_GRAYSCALEALPHA = 4;
```

C++

```
COLOR_GRAYSCALEALPHA = 4;
```

File

pngimage

Description

This is constant pngimage.COLOR_GRAYSCALEALPHA.

56.5.3 pngimage.COLOR_PALETTE Constant

Pascal

```
COLOR_PALETTE = 3;
```

C++

```
COLOR_PALETTE = 3;
```

File

pngimage

Description

This is constant pngimage.COLOR_PALETTE.

56.5.4 pngimage.COLOR_RGB Constant

Pascal

```
COLOR_RGB = 2;
```

C++

```
COLOR_RGB = 2;
```

File

pngimage

Description

This is constant pngimage.COLOR_RGB.

56.5.5 pngimage.COLOR_RGBALPHA Constant

Pascal

```
COLOR_RGBALPHA = 6;
```

C++

```
COLOR_RGBALPHA = 6;
```

File

pngimage

Description

This is constant pngimage.COLOR_RGBALPHA.

56.5.6 pngimage.FILTER_AVERAGE Constant

Pascal

```
FILTER_AVERAGE = 3;
```

C++

```
FILTER_AVERAGE = 3;
```

File

pngimage

Description

This is constant pngimage.FILTER_AVERAGE.

56.5.7 pngimage.FILTER_NONE Constant

Pascal

```
FILTER_NONE = 0;
```

C++

```
FILTER_NONE = 0;
```

File

pngimage

Description

This is constant pngimage.FILTER_NONE.

56.5.8 pngimage.FILTER_PAETH Constant

Pascal

```
FILTER_PAETH = 4;
```

C++

```
FILTER_PAETH = 4;
```

File

pngimage

Description

This is constant pngimage.FILTER_PAETH.

56.5.9 pngimage.FILTER_SUB Constant

Pascal

```
FILTER_SUB = 1;
```

C++

```
FILTER_SUB = 1;
```

File

pngimage

Description

This is constant pngimage.FILTER_SUB.

56.5.10 pngimage.FILTER_UP Constant

Pascal

```
FILTER_UP = 2;
```

C++

```
FILTER_UP = 2;
```

File

pngimage

Description

This is constant pngimage.FILTER_UP.

56.5.11 pngimage.Index Constant

Pascal

```
Index: Integer]: pByteArray read GetAlphaScanline = ;
```

C++

```
Integer]: pByteArray read GetAlphaScanline Index;
```

File

pngimage

Description

This is constant pngimage.Index.

56.5.12 pngimage.LibraryVersion Constant

Pascal

```
LibraryVersion = '1.564';
```

C++

```
LibraryVersion = '1.564';
```

File

pngimage

Description

This is constant pngimage.LibraryVersion.

56.5.13 pngimage.soFromBeginning Constant

Pascal

```
soFromBeginning = 0;
```

C++

```
soFromBeginning = 0;
```

File

pngimage

Description

This is constant pngimage.soFromBeginning.

56.5.14 pngimage.soFromCurrent Constant

Pascal

```
soFromCurrent = 1;
```

C++

```
soFromCurrent = 1;
```

File

pngimage

Description

This is constant pngimage.soFromCurrent.

56.5.15 pngimage.soFromEnd Constant

Pascal

```
soFromEnd = 2;
```

C++

```
soFromEnd = 2;
```

File

pngimage

Description

This is constant pngimage.soFromEnd.

56.5.16 pngimage.Z_FINISH Constant

Pascal

```
Z_FINISH = 4;
```

C++

```
Z_FINISH = 4;
```

File

pngimage

Description

This is constant pngimage.Z_FINISH.

56.5.17 pngimage.Z_NO_FLUSH Constant

Pascal

```
Z_NO_FLUSH = 0;
```

C++

```
Z_NO_FLUSH = 0;
```

File

pngimage

Description

This is constant pngimage.Z_NO_FLUSH.

56.5.18 pngimage.Z_STREAM_END Constant

Pascal

```
Z_STREAM_END = 1;
```

C++

```
Z_STREAM_END = 1;
```

File

pngimage

Description

This is constant pngimage.Z_STREAM_END.

56.5.19 pngimage.ZLIBErrors Constant

Pascal

```
ZLIBErrors: Array[-6..2] of string = ('incompatible version (-6)', 'buffer error (-5)',
'insufficient memory (-4)', 'data error (-3)', 'stream error (-2)', 'file error (-1)', '(0)',
'stream end (1)', 'need dictionary (2)');
```

C++

```
Array[-6..2] of string ZLIBErrors = ('incompatible version (-6)', 'buffer error (-5)',
'insufficient memory (-4)', 'data error (-3)', 'stream error (-2)', 'file error (-1)', '(0)',
'stream end (1)', 'need dictionary (2)');
```

File

pngimage

Description

This is constant pngimage.ZLIBErrors.

57 Printers Namespace

This is namespace Printers.

Topics

Name	Description
Classes (see page 3038)	The following table lists classes in this documentation.
Functions (see page 3066)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 3070)	The following table lists structs, records, enums in this documentation.
Types (see page 3072)	The following table lists types in this documentation.

Classes

	Name	Description
	EPrinter (see page 3038)	EPrinter is the exception class for printing errors.
	TPrinter (see page 3038)	TPrinter encapsulates the Windows printer interface.

Functions

	Name	Description
	AssignPrn (see page 3066)	Assigns a text file variable to the printer.
	Printer (see page 3066)	Returns a global instance of TPrinter (see page 3038) to manage interaction with the printer.
	SetPrinter (see page 3068)	Replaces the global instance of TPrinter (see page 3038) that manages interaction with the printer.

Structs, Records, Enums

	Name	Description
	TPrinterCapability (see page 3070)	TPrinterCapability and TPrinterCapabilities (see page 3072) specify the capabilities of a printer device driver.
	TPrinterOrientation (see page 3070)	TPrinterOrientation indicates the orientation of pages when printing.
	TPrinterState (see page 3071)	This is record Printers.TPrinterState.

Types

Name	Description
TPrinterCapabilities (see page 3072)	TPrinterCapabilities and TPrinterCapability (see page 3070) specify the capabilities of a printer device driver.

57.1 Classes

The following table lists classes in this documentation.

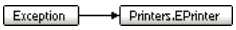
Classes

	Name	Description
	EPrinter (see page 3038)	EPrinter is the exception class for printing errors.
	TPrinter (see page 3038)	TPrinter encapsulates the Windows printer interface.

57.1.1 EPrinter Class

EPrinter is the exception class for printing errors.

Class Hierarchy



Pascal

```
EPrinter = class(Exception);
```

C++

```
class EPrinter : public Exception;
```

File

Printers

Description

EPrinter is raised when a printing error occurs. For example, if an application attempts to use a printer that doesn't exist, or if a job can't be sent to the printer for some other reason, an EPrinter exception is raised.

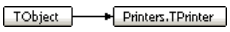
See Also

TPrinter (see page 3038)

57.1.2 TPrinter Class

TPrinter encapsulates the Windows printer interface.

Class Hierarchy



Pascal

```
TPrinter = class(TObject);
```

C++

```
class TPrinter : public TObject;
```

File

Printers

Description

Use TPrinter to manage any printing performed by an application. Obtain an instance of TPrinter by calling the Printer (see page 3066) function in the Printers (see page 3062) unit.

To determine how the printed image of the form appears, use the PrintScale property of a TForm component.

A print job is started by a call to BeginDoc (see page 3043). The application sends commands by rendering through a Text variable or the printer's canvas. You can move to a new page by calling the NewPage (see page 3047) method. The job stays open until the application calls EndDoc (see page 3045). If a problem occurs and you need to terminate a print job that was not sent to the printer successfully, call the Abort (see page 3041) method.

Use the TPrinter properties to configure the print job. For example, the title displayed in the Print Manager (and on network header pages) is determined by the Title (see page 3065) property. Copies (see page 3054) determines the number of copies to print, and Orientation (see page 3056) lets you specify whether to print in portrait or landscape mode.

TPrinter includes several read-only properties as well, that let you determine which page is currently being printed, the fonts available on the printer, the paper size, and so on.

When creating a TPrinter descendant, you must call the SetPrinter (see page 3050) routine in order for the TPrinter descendant object to work correctly.

See Also

Printer (see page 3066)

TCustomForm

SetPrinter (see page 3068)

57.1.2.1 TPrinter Members

The following tables list the members exposed by TPrinter.

Public Methods

	Name	Description
	Abort (see page 3041)	Terminates the printing of a print job, dropping all unprinted data.
	BeginDoc (see page 3043)	Sends a print job to the printer.
	Create (see page 3040)	Creates a printer.
V	Destroy (see page 3040)	Destroys an instance of a printer.
	EndDoc (see page 3045)	Ends the current print job and closes the text file variable.
	GetPrinter (see page 3047)	Retrieves information about the current printer.
	NewPage (see page 3047)	Starts a new page and increments the PageNumber (see page 3059) property.
	Refresh (see page 3049)	Updates the Fonts (see page 3054) and Printers (see page 3062) properties.
	SetPrinter (see page 3050)	Specifies the current printer.

Public Properties

	Name	Description
R	Aborted (see page 3051)	Determines if the user aborted the print job.
R	Canvas (see page 3053)	Represents the surface of the currently printing page.
R	Capabilities (see page 3053)	Indicates the current settings of a printer device driver.
	Copies (see page 3054)	Specifies the number of copies to print.
R	Fonts (see page 3054)	Lists the fonts supported by the current printer.

R	Handle (see page 3054)	Provides access to the handle of the printer object.
	Orientation (see page 3056)	Determines if the print job prints vertically or horizontally on a page.
R	PageHeight (see page 3057)	Indicates the height (in pixels) of the currently printing page.
R	PageNumber (see page 3059)	Indicates the number of the page currently printing.
R	PageWidth (see page 3060)	Indicates the value of width (in pixels) of the currently printing page.
	PrinterIndex (see page 3062)	Specifies which printer listed in the Printers (see page 3062) property is the currently selected printer.
R	Printers (see page 3062)	Lists all printers installed in Windows.
R	Printing (see page 3062)	Indicates whether a print job is printing.
	Title (see page 3065)	Determines the text that is listed in the Print Manager and on network header pages.

57.1.2.2 TPrinter.Create Constructor

Creates a printer.

Pascal

```
constructor Create;
```

C++

```
__fastcall TPrinter();
```

Description

Never call Create directly. Create is called automatically by the Printer (see page 3066) function. Create allocates memory for a printer and calls the inherited Create. Then it sets the correct driver, device, and port.

See Also

System::TObject::Create

57.1.2.3 TPrinter.Destroy Destructor

Destroys an instance of a printer.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TPrinter();
```

Description

Do not call Destroy directly. The printer object is automatically created and destroyed.

When Destroy is called, it verifies that any currently printing job has ended, then frees resources such as the handle, fonts, and canvas, closes the printer, and calls the inherited Destroy method.

See Also

System::TObject::Destroy

System::TObject::Free

57.1.2.4 TPrinter Methods

The methods of the TPrinter class are listed here.

Public Methods

	Name	Description
	Abort (see page 3041)	Terminates the printing of a print job, dropping all unprinted data.
	BeginDoc (see page 3043)	Sends a print job to the printer.
	EndDoc (see page 3045)	Ends the current print job and closes the text file variable.
	GetPrinter (see page 3047)	Retrieves information about the current printer.
	NewPage (see page 3047)	Starts a new page and increments the PageNumber (see page 3059) property.
	Refresh (see page 3049)	Updates the Fonts (see page 3054) and Printers (see page 3062) properties.
	SetPrinter (see page 3050)	Specifies the current printer.

57.1.2.4.1 TPrinter.Abort Method

Terminates the printing of a print job, dropping all unprinted data.

Pascal

```
procedure Abort;
```

C++

```
__fastcall Abort();
```

Description

Use Abort to terminate the print job before it completes; otherwise, use the EndDoc (see page 3045) method.

When Abort is called, the device is then set for the next print job.

C++ Examples:

```
/*
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
*/
#include <Printers.hpp>

void __fastcall TForm1::FormCreate(TObject *Sender)
{
   KeyPreview = True;

}

void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)

{
  if (Key == VK_ESCAPE && Printer()->Printing)
  {
    Printer()->Abort();
    MessageDlg("Printing aborted", mtInformation,
               TMsgDlgButtons() << mbOK,0);
  }
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMemo *Memo1 = new TMemo(Form1); // The parent (Form1) will clean up this object.
  Memo1->Parent = Form1;
  Memo1->Visible = True;
  Memo1->Width = 700;
  if (OpenDialog1->Execute())
  {
    Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
```

```
    Printer()->BeginDoc();
    int X = 200;
    int Y = 200;
    for (int I = 0; I <= 10; I++)
      if (!Printer()->Aborted)
      {
        Printer()->Canvas->TextOut(X, Y, Memo1->Lines->Strings[I]);
        Y = Y + 80;
        if (Y > (Printer()->PageHeight - 300))
        {
          Y = 200;
          Printer()->NewPage();
          Sleep(1000);  // to give you time to abort!
        }
      }
    if (!Printer()->Aborted) Printer()->EndDoc();
  }
}
```

Delphi Examples:

```
{
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
}

uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  I, X, Y: Integer;
  Memo1 : TMemo;
  r: TRect;
begin
  Memo1 := TMemo.Create(Form1);
  Memo1.Parent := Form1;
  Memo1.Visible := True;
  Memo1.Width := 700;
  if (OpenDialog1.Execute) then
  begin
    Memo1.Lines.LoadFromFile(OpenDialog1.FileName);

    Printer.BeginDoc;
    X := 200;
    Y := 200;
    for I := 0 to 140 do
      if (not Printer.Aborted) then
      begin
        Printer.Canvas.TextOut(X, Y, Memo1.Lines[I]);
        Y := Y + 80;
        if (Y > (Printer.PageHeight - 300)) then
        begin
          Y := 200;
          Printer.NewPage;
          Sleep(1000);  // to give you time to abort!
        end;
      end;
    if (not Printer.Aborted) then Printer.EndDoc;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
   KeyPreview := True;
end;
```

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
if (Key=VK_ESCAPE) and Printer.Printing then
  begin
  Printer.Abort;
  MessageDlg('Printing aborted', mtInformation, [mbOK],0);
  end;
end;
```

See Also

Aborted (see page 3051)

BeginDoc (see page 3043)

EndDoc (see page 3045)

Printing (see page 3062)

57.1.2.4.2 TPrinter.BeginDoc Method

Sends a print job to the printer.

Pascal

```
procedure BeginDoc;
```

C++

```
__fastcall BeginDoc();
```

Description

Call BeginDoc to initiate a print job. If the print job is sent successfully, the application calls EndDoc (see page 3045) to end the print job. The print job won't actually start printing until EndDoc (see page 3045) is called.

C++ Examples:

```
{
This example uses a button, a Page Control, and a Print
dialog box on a form. When the user clicks the button, the
print dialog is displayed.  The user can select any subset
of the pages in the page control for printing.  The selected
pages are then printed.  To run this example successfully,
you must add the Printers unit to the uses clause of your
unit.  Right click on the PageControl to add at least one
page to the PageControl.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  PrintDialog1->Options.Clear();
  PrintDialog1->Options << poPageNums << poSelection;
  PrintDialog1->FromPage = 1;
  PrintDialog1->MinPage = 1;
  PrintDialog1->ToPage = PageControl1->PageCount;
  PrintDialog1->MaxPage = PageControl1->PageCount;
  if (PrintDialog1->Execute())
  {
    int Start, Stop;
    // determine the range the user wants to print
    switch (PrintDialog1->PrintRange)
    {
      case prSelection:
        Start = PageControl1->ActivePage->PageIndex;
        Stop = Start;
        break;
```

```
      case prPageNums:
        Start = PrintDialog1->FromPage - 1;
        Stop =  PrintDialog1->ToPage - 1;
        break;
      default:  // prAllPages
        Start = PrintDialog1->MinPage - 1;
        Stop = PrintDialog1->MaxPage - 1;
        break;
    }
    // now, print the pages
    Printer()->BeginDoc();
    for (int i = Start; i <= Stop; i++)
    {
      PageControl1->Pages[i]->PaintTo(Printer()->Handle, 10, 10);
      if (i != Stop)
        Printer()->NewPage();
    }
    Printer()->EndDoc();
  }
}
```

Delphi Examples:

```
{
This example uses a button, a Page Control, and a Print
dialog box on a form. When the user clicks the button, the
print dialog is displayed.  The user can select any subset
of the pages in the page control for printing.  The selected
pages are then printed.  To run this example successfully,
you must add the Printers unit to the uses clause of your
unit.  Right click on the PageControl to add at least one
page to the PageControl.
}
procedure TForm1.Button1Click(Sender:TObject);
var
  I, Start, Stop: Integer;
begin
PrintDialog1.Options := [poPageNums, poSelection];
PrintDialog1.FromPage := 1;
PrintDialog1.MinPage := 1;
PrintDialog1.ToPage := PageControl1.PageCount;
PrintDialog1.MaxPage := PageControl1.PageCount;

if PrintDialog1.Execute() then
  begin
    { determine the range the user wants to print }
    with PrintDialog1 do
    begin
      if PrintRange = prAllPages then
        begin
        Start := MinPage - 1;
        Stop := MaxPage - 1;
      end
      else if PrintRange = prSelection then
      begin
        Start := PageControl1.ActivePage.PageIndex;
        Stop := Start;
        end
      else  { PrintRange = prPageNums }
      begin
        Start := FromPage - 1;
        Stop := ToPage - 1;
      end;
    end;
    { now, print the pages }
    with Printer do
    begin
```

```
        BeginDoc;
        for I := Start to Stop do
        begin
          PageControl1.Pages[I].PaintTo(Handle, 10, 10);
          if I <> Stop then
            NewPage;
        end;
        EndDoc;
      end;
    end;
end;
```

See Also

Abort (see page 3041)

EndDoc (see page 3045)

Printing (see page 3062)

57.1.2.4.3 TPrinter.EndDoc Method

Ends the current print job and closes the text file variable.

Pascal

```
procedure EndDoc;
```

C++

```
__fastcall EndDoc();
```

Description

EndDoc terminates the print job (and closes the currently open Text). The print job will begin printing on the printer after a call to EndDoc.

After the application calls EndDoc, the printer begins printing. Use EndDoc after successfully sending a print job to the printer. If the print job isn't successful, use the Abort (see page 3041) method.

C++ Examples:

```
{
This example uses a button, a Page Control, and a Print
dialog box on a form. When the user clicks the button, the
print dialog is displayed.  The user can select any subset
of the pages in the page control for printing.  The selected
pages are then printed.  To run this example successfully,
you must add the Printers unit to the uses clause of your
unit.  Right click on the PageControl to add at least one
page to the PageControl.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  PrintDialog1->Options.Clear();
  PrintDialog1->Options << poPageNums << poSelection;
  PrintDialog1->FromPage = 1;
  PrintDialog1->MinPage = 1;
  PrintDialog1->ToPage = PageControl1->PageCount;
  PrintDialog1->MaxPage = PageControl1->PageCount;
  if (PrintDialog1->Execute())
  {
    int Start, Stop;
    // determine the range the user wants to print
    switch (PrintDialog1->PrintRange)
    {
      case prSelection:
```

```
        Start = PageControl1->ActivePage->PageIndex;
        Stop = Start;
        break;
      case prPageNums:
        Start = PrintDialog1->FromPage - 1;
        Stop =  PrintDialog1->ToPage - 1;
        break;
      default:  // prAllPages
        Start = PrintDialog1->MinPage - 1;
        Stop = PrintDialog1->MaxPage - 1;
        break;
    }
    // now, print the pages
    Printer()->BeginDoc();
    for (int i = Start; i <= Stop; i++)
    {
      PageControl1->Pages[i]->PaintTo(Printer()->Handle, 10, 10);
      if (i != Stop)
        Printer()->NewPage();
    }
    Printer()->EndDoc();
  }
}
```

Delphi Examples:

```
{
This example uses a button, a Page Control, and a Print
dialog box on a form. When the user clicks the button, the
print dialog is displayed.  The user can select any subset
of the pages in the page control for printing.  The selected
pages are then printed.  To run this example successfully,
you must add the Printers unit to the uses clause of your
unit.  Right click on the PageControl to add at least one
page to the PageControl.
}
procedure TForm1.Button1Click(Sender:TObject);
var
  I, Start, Stop: Integer;
begin
PrintDialog1.Options := [poPageNums, poSelection];
PrintDialog1.FromPage := 1;
PrintDialog1.MinPage := 1;
PrintDialog1.ToPage := PageControl1.PageCount;
PrintDialog1.MaxPage := PageControl1.PageCount;

if PrintDialog1.Execute() then
  begin
    { determine the range the user wants to print }
    with PrintDialog1 do
    begin
      if PrintRange = prAllPages then
        begin
        Start := MinPage - 1;
        Stop := MaxPage - 1;
      end
      else if PrintRange = prSelection then
      begin
        Start := PageControl1.ActivePage.PageIndex;
        Stop := Start;
        end
      else  { PrintRange = prPageNums }
      begin
        Start := FromPage - 1;
        Stop := ToPage - 1;
      end;
    end;
```

```
      { now, print the pages }
      with Printer do
      begin
        BeginDoc;
        for I := Start to Stop do
        begin
          PageControl1.Pages[I].PaintTo(Handle, 10, 10);
          if I <> Stop then
            NewPage;
        end;
        EndDoc;
      end;
    end;
  end;
```

See Also

Abort (see page 3041)

BeginDoc (see page 3043)

Printing (see page 3062)

57.1.2.4.4 TPrinter.GetPrinter Method

Retrieves information about the current printer.

Pascal

```
procedure GetPrinter(ADevice: PChar; ADriver: PChar; APort: PChar; var ADeviceMode: THandle);
```

C++

```
__fastcall GetPrinter(const char * ADevice, const char * ADriver, const char * APort, THandle
ADeviceMode);
```

Description

Call GetPrinter to obtain information about the current printer.

ADevice is a buffer that receives the device name of the current printer. This is the name that appears in the Print Manager (for example, "Epson FX-80"). It is not the printer model name.

ADriver is a buffer that receives the driver name for the current printer. Some versions of Windows do not use printer driver names, in which case this buffer is unused. On versions of Windows that support printer driver names, this is the name of a printer driver (usually 'WINSPOOL').

APort is a buffer that receives the port used by the current printer. It is not used on 32-bit versions of Windows, but is included for backward compatibility.

DeviceMode returns the handle to a memory buffer containing device-specific initialization data for the device driver. Use the GlobalLock function to convert this handle into a pointer to a DeviceMode record (Delphi) or DEVMODE structure (C++).

See Also

Printers (see page 3062)

57.1.2.4.5 TPrinter.NewPage Method

Starts a new page and increments the PageNumber (see page 3059) property.

Pascal

```
procedure NewPage;
```

C++

```
__fastcall NewPage();
```

Description

NewPage forces the current print job to begin printing on a new page in the printer.

NewPage increments the value of the PageNumber (see page 3059) property. It also resets the value of the Pen property of the Canvas (see page 3053) back to (0, 0).

C++ Examples:

```
{
This example uses a button, a Page Control, and a Print
dialog box on a form. When the user clicks the button, the
print dialog is displayed.  The user can select any subset
of the pages in the page control for printing.  The selected
pages are then printed.  To run this example successfully,
you must add the Printers unit to the uses clause of your
unit.  Right click on the PageControl to add at least one
page to the PageControl.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  PrintDialog1->Options.Clear();
  PrintDialog1->Options << poPageNums << poSelection;
  PrintDialog1->FromPage = 1;
  PrintDialog1->MinPage = 1;
  PrintDialog1->ToPage = PageControl1->PageCount;
  PrintDialog1->MaxPage = PageControl1->PageCount;
  if (PrintDialog1->Execute())
  {
    int Start, Stop;
    // determine the range the user wants to print
    switch (PrintDialog1->PrintRange)
    {
      case prSelection:
        Start = PageControl1->ActivePage->PageIndex;
        Stop = Start;
        break;
      case prPageNums:
        Start = PrintDialog1->FromPage - 1;
        Stop =  PrintDialog1->ToPage - 1;
        break;
      default:  // prAllPages
        Start = PrintDialog1->MinPage - 1;
        Stop = PrintDialog1->MaxPage - 1;
        break;
    }
    // now, print the pages
    Printer()->BeginDoc();
    for (int i = Start; i <= Stop; i++)
    {
      PageControl1->Pages[i]->PaintTo(Printer()->Handle, 10, 10);
      if (i != Stop)
        Printer()->NewPage();
    }
    Printer()->EndDoc();
  }
}
```

Delphi Examples:

```
{
This example uses a button, a Page Control, and a Print
dialog box on a form. When the user clicks the button, the
```

```
  print dialog is displayed.  The user can select any subset
  of the pages in the page control for printing.  The selected
  pages are then printed.  To run this example successfully,
  you must add the Printers unit to the uses clause of your
  unit.  Right click on the PageControl to add at least one
  page to the PageControl.
  }
  procedure TForm1.Button1Click(Sender:TObject);
  var
    I, Start, Stop: Integer;
  begin
  PrintDialog1.Options := [poPageNums, poSelection];
  PrintDialog1.FromPage := 1;
  PrintDialog1.MinPage := 1;
  PrintDialog1.ToPage := PageControl1.PageCount;
  PrintDialog1.MaxPage := PageControl1.PageCount;

  if PrintDialog1.Execute() then
    begin
      { determine the range the user wants to print }
      with PrintDialog1 do
      begin
        if PrintRange = prAllPages then
          begin
          Start := MinPage - 1;
          Stop := MaxPage - 1;
        end
        else if PrintRange = prSelection then
        begin
          Start := PageControl1.ActivePage.PageIndex;
          Stop := Start;
          end
        else  { PrintRange = prPageNums }
        begin
          Start := FromPage - 1;
          Stop := ToPage - 1;
        end;
      end;
      { now, print the pages }
      with Printer do
      begin
        BeginDoc;
        for I := Start to Stop do
        begin
          PageControl1.Pages[I].PaintTo(Handle, 10, 10);
          if I <> Stop then
            NewPage;
        end;
        EndDoc;
      end;
    end;
  end;
```

See Also

PageNumber (see page 3059)

57.1.2.4.6 TPrinter.Refresh Method

Updates the Fonts (see page 3054) and Printers (see page 3062) properties.

Pascal

```
procedure Refresh;
```

C++

```
__fastcall Refresh();
```

Description

Call Refresh when the installed fonts or printers on the current system may have changed. Refresh clears the internal cache of installed fonts and printers, so that the Fonts (see page 3054) and Printers (see page 3062) properties are updated when they are next read.

See Also

Fonts (see page 3054)

Printers (see page 3062)

57.1.2.4.7 TPrinter.SetPrinter Method

Specifies the current printer.

Pascal

```
procedure SetPrinter(ADevice: PChar; ADriver: PChar; APort: PChar; ADeviceMode: THandle);
```

C++

```
__fastcall SetPrinter(const char * ADevice, const char * ADriver, const char * APort, THandle
ADeviceMode);
```

Description

Call SetPrinter to change the current printer.

ADevice specifies the device name of the new printer. This is the name that appears in the Print Manager (for example, "Epson FX-80"). It is not the printer model name.

ADriver specifies the driver name for the new printer (usually 'WINSPOOL').

APort is included for backward compatibility. It should be set to an empty string.

DeviceMode is the handle to a memory buffer device-specific initialization data for the device driver. This handle can be obtained by calling the GlobalAlloc function to allocate a DeviceMode record (Delphi) or DEVMODE structure (C++).

See Also

Printers (see page 3062)

57.1.2.5 TPrinter Properties

The properties of the TPrinter class are listed here.

Public Properties

	Name	Description
R	Aborted (see page 3051)	Determines if the user aborted the print job.
R	Canvas (see page 3053)	Represents the surface of the currently printing page.
R	Capabilities (see page 3053)	Indicates the current settings of a printer device driver.
	Copies (see page 3054)	Specifies the number of copies to print.
R	Fonts (see page 3054)	Lists the fonts supported by the current printer.
R	Handle (see page 3054)	Provides access to the handle of the printer object.
	Orientation (see page 3056)	Determines if the print job prints vertically or horizontally on a page.
R	PageHeight (see page 3057)	Indicates the height (in pixels) of the currently printing page.

R	PageNumber (see page 3059)	Indicates the number of the page currently printing.
R	PageWidth (see page 3060)	Indicates the value of width (in pixels) of the currently printing page.
	PrinterIndex (see page 3062)	Specifies which printer listed in the Printers (see page 3062) property is the currently selected printer.
R	Printers (see page 3062)	Lists all printers installed in Windows.
R	Printing (see page 3062)	Indicates whether a print job is printing.
	Title (see page 3065)	Determines the text that is listed in the Print Manager and on network header pages.

57.1.2.5.1 TPrinter.Aborted Property

Determines if the user aborted the print job.

Pascal

```
property Aborted: Boolean;
```

C++

```
__property Boolean Aborted;
```

Description

If Aborted is true, the print job was aborted. If it is false, the user did not abort the print job.

The Abort (see page 3041) method sets the Aborted property to true.

C++ Examples:

```
/*
The following code displays a dialog box if the print job
was aborted.  This code aborts a print job if the user
presses Esc. Note that you should set KeyPreview to true
to ensure that the OnKeyDown event handler of Form1 is called.
Also notice that the Button1Click routine continues to run
even after the print job has been aborted.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMemo *Memo1 = new TMemo(Form1); // The parent (Form1) will clean up this object.
  Memo1->Parent = Form1;
  Memo1->Visible = True;
  Memo1->Width = 700;
  if (OpenDialog1->Execute())
  {
    Memo1->Lines->LoadFromFile(OpenDialog1->FileName);

    Printer()->BeginDoc();
    int X = 200;
    int Y = 200;
    for (int I = 0; I <= 140; I++)
      if (!Printer()->Aborted)
      {
        Printer()->Canvas->TextOut(X, Y, Memo1->Lines->Strings[I]);
        Y = Y + 80;
        if (Y > (Printer()->PageHeight - 300))
        {
          Y = 200;
          Printer()->NewPage();
          Sleep(1000);  // to give you time to abort!
        }
      }
    if (!Printer()->Aborted) Printer()->EndDoc();
  }
  if (Printer()->Aborted)
    MessageDlg(
```

```
      "The print job did not finish printing",
      mtInformation, TMsgDlgButtons() << mbOK, 0);
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  KeyPreview = True;
}

void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)

{
  if (Key == VK_ESCAPE && Printer()->Printing)
  {
    Printer()->Abort();
    MessageDlg("Printing aborted", mtInformation,
               TMsgDlgButtons() << mbOK,0);
  }

}
```

Delphi Examples:

```
{
The following code displays a dialog box if the print job
was aborted.  This code aborts a print job if the user
presses Esc. Note that you should set KeyPreview to true
to ensure that the OnKeyDown event handler of Form1 is called.
Also notice that the Button1Click routine continues to run
even after the print job has been aborted.
}

uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  I, X, Y: Integer;
  Memo1 : TMemo;
  r: TRect;
begin
  Memo1 := TMemo.Create(Form1);
  Memo1.Parent := Form1;
  Memo1.Visible := True;
  Memo1.Width := 700;
  if (OpenDialog1.Execute) then
  begin
    Memo1.Lines.LoadFromFile(OpenDialog1.FileName);

    Printer.BeginDoc;
    X := 200;
    Y := 200;
    for I := 0 to 140 do
      if (not Printer.Aborted) then
      begin
        Printer.Canvas.TextOut(X, Y, Memo1.Lines[I]);
        Y := Y + 80;
        if (Y > (Printer.PageHeight - 300)) then
        begin
          Y := 200;
          Printer.NewPage;
          Sleep(1000);  // to give you time to abort!
        end;
      end;
    if (not Printer.Aborted) then Printer.EndDoc;
  end;
  if Printer.Aborted then
    MessageDlg('The print job did not finish printing', mtInformation, [mbOK], 0);
```

```
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
   KeyPreview := True;
end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
if (Key=VK_ESCAPE) and Printer.Printing then
  begin
  Printer.Abort;
//  MessageDlg('Printing aborted', mtInformation, [mbOK],0);
  end;
end;
```

See Also

Abort (see page 3041)

57.1.2.5.2 TPrinter.Canvas Property

Represents the surface of the currently printing page.

Pascal

```
property Canvas: TCanvas;
```

C++

```
__property TCanvas Canvas;
```

Description

Canvas represents the printing surface of a page. Use the Brush, Font, and Pen properties of the Canvas object to determine how drawing or text appears on the page.

Note:

See Also

Brush

CopyRect

Draw

Font

Pen

StretchDraw

Handle

57.1.2.5.3 TPrinter.Capabilities Property

Indicates the current settings of a printer device driver.

Pascal

```
property Capabilities: TPrinterCapabilities;
```

C++

```
__property TPrinterCapabilities Capabilities;
```

Description

Capabilities contains the capabilities for the printer, indicating the orientation, number of copies, and whether or not collated is selected.

57.1.2.5.4 TPrinter.Copies Property

Specifies the number of copies to print.

Pascal

```
property Copies: Integer;
```

C++

```
__property int Copies;
```

Description

Set Copies to indicate the number of copies to print.

Copies uses the device mode to determine the number of copies to print.

57.1.2.5.5 TPrinter.Fonts Property

Lists the fonts supported by the current printer.

Pascal

```
property Fonts: TStrings;
```

C++

```
__property TStrings Fonts;
```

Description

Read Fonts to get a list of the fonts supported by the printer. Each string in Fonts is the name of a typeface.

Fonts contains TrueType fonts even if the printer doesn't support them natively because the Windows Graphics Device Interface (GDI) can draw TrueType fonts accurately when a print job uses them.

See Also

Name

57.1.2.5.6 TPrinter.Handle Property

Provides access to the handle of the printer object.

Pascal

```
property Handle: HDC;
```

C++

```
__property HDC Handle;
```

Description

Use Handle to gain access to the handle of the printer object.

C++ Examples:

```
{
This example uses a button, a Page Control, and a Print
```

```
dialog box on a form. When the user clicks the button, the
print dialog is displayed.  The user can select any subset
of the pages in the page control for printing.  The selected
pages are then printed.  To run this example successfully,
you must add the Printers unit to the uses clause of your
unit.  Right click on the PageControl to add at least one
page to the PageControl.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  PrintDialog1->Options.Clear();
  PrintDialog1->Options << poPageNums << poSelection;
  PrintDialog1->FromPage = 1;
  PrintDialog1->MinPage = 1;
  PrintDialog1->ToPage = PageControl1->PageCount;
  PrintDialog1->MaxPage = PageControl1->PageCount;
  if (PrintDialog1->Execute())
  {
    int Start, Stop;
    // determine the range the user wants to print
    switch (PrintDialog1->PrintRange)
    {
      case prSelection:
        Start = PageControl1->ActivePage->PageIndex;
        Stop = Start;
        break;
      case prPageNums:
        Start = PrintDialog1->FromPage - 1;
        Stop =  PrintDialog1->ToPage - 1;
        break;
      default:  // prAllPages
        Start = PrintDialog1->MinPage - 1;
        Stop = PrintDialog1->MaxPage - 1;
        break;
    }
    // now, print the pages
    Printer()->BeginDoc();
    for (int i = Start; i <= Stop; i++)
    {
      PageControl1->Pages[i]->PaintTo(Printer()->Handle, 10, 10);
      if (i != Stop)
        Printer()->NewPage();
    }
    Printer()->EndDoc();
  }
}
```

Delphi Examples:

```
{
This example uses a button, a Page Control, and a Print
dialog box on a form. When the user clicks the button, the
print dialog is displayed.  The user can select any subset
of the pages in the page control for printing.  The selected
pages are then printed.  To run this example successfully,
you must add the Printers unit to the uses clause of your
unit.  Right click on the PageControl to add at least one
page to the PageControl.
}
procedure TForm1.Button1Click(Sender:TObject);
var
  I, Start, Stop: Integer;
begin
PrintDialog1.Options := [poPageNums, poSelection];
PrintDialog1.FromPage := 1;
PrintDialog1.MinPage := 1;
PrintDialog1.ToPage := PageControl1.PageCount;
```

```
PrintDialog1.MaxPage := PageControl1.PageCount;

if PrintDialog1.Execute() then
  begin
    { determine the range the user wants to print }
    with PrintDialog1 do
    begin
      if PrintRange = prAllPages then
        begin
        Start := MinPage - 1;
        Stop := MaxPage - 1;
      end
      else if PrintRange = prSelection then
      begin
        Start := PageControl1.ActivePage.PageIndex;
        Stop := Start;
        end
      else  { PrintRange = prPageNums }
      begin
        Start := FromPage - 1;
        Stop := ToPage - 1;
      end;
    end;
    { now, print the pages }
    with Printer do
    begin
      BeginDoc;
      for I := Start to Stop do
      begin
        PageControl1.Pages[I].PaintTo(Handle, 10, 10);
        if I <> Stop then
          NewPage;
      end;
      EndDoc;
    end;
  end;
end;
```

57.1.2.5.7 TPrinter.Orientation Property

Determines if the print job prints vertically or horizontally on a page.

Pascal

```
property Orientation: TPrinterOrientation;
```

C++

```
__property TPrinterOrientation Orientation;
```

Description

Use Orientation to determine if a print job prints in landscape or portrait.

C++ Examples:

```
/*
This example uses two radio buttons on a form named
Landscape and Portrait. The form also includes a button.
When the user selects an orientation by clicking one of the
radio buttons and then clicks the button to print one line
of text, the print job prints using the selected orientation:
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Printer()->BeginDoc();
  Printer()->Canvas->TextOut(100,100,"Hi there!");
```

```
  Printer()->EndDoc();
}

void __fastcall TForm1::PortraitClick(TObject *Sender)
{
  Printer()->Orientation = poPortrait;
}

void __fastcall TForm1::LandscapeClick(TObject *Sender)
{
  Printer()->Orientation = poLandscape;
}
```

Delphi Examples:

```
{
This example uses two radio buttons on a form named
Landscape and Portrait. The form also includes a button.
When the user selects an orientation by clicking one of the
radio buttons and then clicks the button to print one line
of text, the print job prints using the selected orientation:
}
uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
begin
  Printer.BeginDoc;
  Printer.Canvas.TextOut(100,100,'Hi there');
  Printer.EndDoc;
end;

procedure TForm1.LandscapeClick(Sender: TObject);
begin
  Printer.Orientation := poLandscape;
end;

procedure TForm1.PortraitClick(Sender: TObject);
begin
  Printer.Orientation := poPortrait;
end;
```

57.1.2.5.8 TPrinter.PageHeight Property

Indicates the height (in pixels) of the currently printing page.

Pascal

```
property PageHeight: Integer;
```

C++

```
__property int PageHeight;
```

Description

Use PageHeight to find the height in pixels of the currently printing page.

C++ Examples:

```
/*
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
*/
#include <Printers.hpp>

void __fastcall TForm1::FormCreate(TObject *Sender)
{
```

```
    KeyPreview = True;

}

void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)

{
  if (Key == VK_ESCAPE && Printer()->Printing)
  {
    Printer()->Abort();
    MessageDlg("Printing aborted", mtInformation,
               TMsgDlgButtons() << mbOK,0);
  }
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMemo *Memo1 = new TMemo(Form1); // The parent (Form1) will clean up this object.
  Memo1->Parent = Form1;
  Memo1->Visible = True;
  Memo1->Width = 700;
  if (OpenDialog1->Execute())
  {
    Memo1->Lines->LoadFromFile(OpenDialog1->FileName);

    Printer()->BeginDoc();
    int X = 200;
    int Y = 200;
    for (int I = 0; I <= 10; I++)
      if (!Printer()->Aborted)
      {
        Printer()->Canvas->TextOut(X, Y, Memo1->Lines->Strings[I]);
        Y = Y + 80;
        if (Y > (Printer()->PageHeight - 300))
        {
          Y = 200;
          Printer()->NewPage();
          Sleep(1000);  // to give you time to abort!
        }
      }
    if (!Printer()->Aborted) Printer()->EndDoc();
  }
}
```

Delphi Examples:

```
{
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
}

uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  I, X, Y: Integer;
  Memo1 : TMemo;
  r: TRect;
begin
  Memo1 := TMemo.Create(Form1);
  Memo1.Parent := Form1;
  Memo1.Visible := True;
  Memo1.Width := 700;
  if (OpenDialog1.Execute) then
  begin
    Memo1.Lines.LoadFromFile(OpenDialog1.FileName);
```

```
    Printer.BeginDoc;
    X := 200;
    Y := 200;
    for I := 0 to 140 do
      if (not Printer.Aborted) then
      begin
        Printer.Canvas.TextOut(X, Y, Memo1.Lines[I]);
        Y := Y + 80;
        if (Y > (Printer.PageHeight - 300)) then
        begin
          Y := 200;
          Printer.NewPage;
          Sleep(1000);  // to give you time to abort!
        end;
      end;
    if (not Printer.Aborted) then Printer.EndDoc;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
   KeyPreview := True;
end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
if (Key=VK_ESCAPE) and Printer.Printing then
  begin
  Printer.Abort;
  MessageDlg('Printing aborted', mtInformation, [mbOK],0);
  end;
end;
```

See Also

PageNumber (see page 3059)

PageWidth (see page 3060)

57.1.2.5.9 TPrinter.PageNumber Property

Indicates the number of the page currently printing.

Pascal

```
property PageNumber: Integer;
```

C++

```
__property int PageNumber;
```

Description

Use PageNumber to find the page number of the currently printing page.

PageNumber is incremented whenever the NewPage (see page 3047) method is called. It is also incremented when a Text variable is written and a CR is encountered on the last line of the page.

C++ Examples:

```
/*
This example uses a button and a status bar on a form. When
the user clicks the button, one line of text is printed on
six separate pages. As each page is printed, a message
indicating the number of the page being printed appears on
the status bar.  Notice that the SimplePanel property of the
```

```
status bar to must be set to true.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  StatusBar1->SimplePanel = True; // so that SimpleText will work
  Printer()->BeginDoc();
  for (int i = 1;i < 7;i++)
  {
    Printer()->Canvas->TextOut(100, 100, "C++Builder rules!");
    StatusBar1->SimpleText = AnsiString("Printing page ") + Printer()->PageNumber;
    Printer()->NewPage();
  }
  Printer()->EndDoc();
}
```

Delphi Examples:

```
{
This example uses a button and a status bar on a form. When
the user clicks the button, one line of text is printed on
six separate pages. As each page is printed, a message
indicating the number of the page being printed appears on
the status bar.  To run this example successfully, you must
add Printers to the uses clause of your unit and set the
SimplePanel property of the status bar to true.
}
uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
begin
  StatusBar1.SimplePanel := True; { so that SimpleText will work }
  Printer.BeginDoc;
  for I := 1 to 6 do
  begin
    Printer.Canvas.TextOut(100, 100, 'Object Pascal is great');
    StatusBar1.SimpleText := 'Now printing page ' + IntToStr(Printer.PageNumber);
    Printer.NewPage;
  end;
  Printer.EndDoc;

end;
```

See Also

NewPage (see page 3047)

PageHeight (see page 3057)

PageWidth (see page 3060)

57.1.2.5.10 TPrinter.PageWidth Property

Indicates the value of width (in pixels) of the currently printing page.

Pascal

```
property PageWidth: Integer;
```

C++

```
__property int PageWidth;
```

Description

Use PageWidth to find the width in pixels of the currently printing page.

C++ Examples:

```
/*
This example uses a button and a memo on a form. When the
user clicks the button, the content of the memo is printed,
with a 200-pixel border around the page.
Note: The TMemo will wrap the lines of the text files, so to
see the TextRect clip, the memo width must be large and the
lines in the text file must be long.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  Memo1->Width = 1000;
  Memo1->Lines->LoadFromFile("..\\readme.txt");
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Printer()->BeginDoc();
  int margin = 1000;
  int X = margin;
  int Y = margin;
  int I = 0;
  while(Y < Printer()->PageHeight)
  {
    Canvas->TextRect(Rect(X, Y, Printer()->PageWidth-margin, Printer()->PageHeight-margin),
                     X, Y, Memo1->Lines->Strings[I]);
    Y = Y + 100;
    I = I + 1;
  }
  Printer()->EndDoc();
}
```

Delphi Examples:

```
{
This example uses a button and a memo on a form. When the
user clicks the button, the content of the memo is printed,
with a 200-pixel border around the page. To run this example
successfully, include <Printers.hpp> in your unit file.
Note: The TMemo will wrap the lines of the text files, so to
see the TextRect clip, the memo width must be large and the
lines in the text file must be long.
}
uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  X, Y, I, margin: Integer;
begin
  with Printer do
  begin
    BeginDoc;
    margin := 1000;
    X := margin;
    Y := margin;
    I := 0;
    while(Y < PageHeight) do
    begin
      Canvas.TextRect(Rect(X, Y, PageWidth-margin, PageHeight-margin),
                      X, Y, Memo1.Lines[I]);
      Y := Y + 100;
      I := I + 1;
    end;
    EndDoc;
  end;
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Memo1.Width := 1000;
  Memo1.Lines.LoadFromFile('readme.txt');
end;
```

See Also

PageHeight (see page 3057)

PageNumber (see page 3059)

57.1.2.5.11 TPrinter.PrinterIndex Property

Specifies which printer listed in the Printers (see page 3062) property is the currently selected printer.

Pascal

```
property PrinterIndex: Integer;
```

C++

```
__property int PrinterIndex;
```

Description

The value of PrinterIndex is the currently selected printer. If this value is changed, EndDoc (see page 3045) is called automatically. To select the default printer, set the value of PrinterIndex to -1.

The list of installed printers is found in the Printers (see page 3062) property. The list of fonts supported by the current printer is found in the Fonts (see page 3054) property.

See Also

EndDoc (see page 3045)

Fonts (see page 3054)

Printers (see page 3062)

57.1.2.5.12 TPrinter.Printers Property

Lists all printers installed in Windows.

Pascal

```
property Printers: TStrings;
```

C++

```
__property TStrings Printers;
```

Description

The list of installed printers is found in the Printers property. The value of the PrinterIndex (see page 3062) property is the currently selected printer. The list of fonts supported by the current printer is found in the Fonts (see page 3054) property.

See Also

GetPrinter (see page 3047)

57.1.2.5.13 TPrinter.Printing Property

Indicates whether a print job is printing.

Pascal

```
property Printing: Boolean;
```

C++

```
__property Boolean Printing;
```

Description

Use Printing to determine if a job is printing.

Printing is true when the application has called the BeginDoc (see page 3043) method but the EndDoc (see page 3045) method (or the Abort (see page 3041) method) hasn't yet been called.

C++ Examples:

```
/*
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
*/
#include <Printers.hpp>

void __fastcall TForm1::FormCreate(TObject *Sender)
{
   KeyPreview = True;

}

void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)

{
  if (Key == VK_ESCAPE && Printer()->Printing)
  {
    Printer()->Abort();
    MessageDlg("Printing aborted", mtInformation,
               TMsgDlgButtons() << mbOK,0);
  }
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMemo *Memo1 = new TMemo(Form1); // The parent (Form1) will clean up this object.
  Memo1->Parent = Form1;
  Memo1->Visible = True;
  Memo1->Width = 700;
  if (OpenDialog1->Execute())
  {
    Memo1->Lines->LoadFromFile(OpenDialog1->FileName);

    Printer()->BeginDoc();
    int X = 200;
    int Y = 200;
    for (int I = 0; I <= 10; I++)
      if (!Printer()->Aborted)
      {
        Printer()->Canvas->TextOut(X, Y, Memo1->Lines->Strings[I]);
        Y = Y + 80;
        if (Y > (Printer()->PageHeight - 300))
        {
          Y = 200;
          Printer()->NewPage();
          Sleep(1000);  // to give you time to abort!
        }
      }
    if (!Printer()->Aborted) Printer()->EndDoc();
```

```
  }
}
```

Delphi Examples:

```
{
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
}

uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  I, X, Y: Integer;
  Memo1 : TMemo;
  r: TRect;
begin
  Memo1 := TMemo.Create(Form1);
  Memo1.Parent := Form1;
  Memo1.Visible := True;
  Memo1.Width := 700;
  if (OpenDialog1.Execute) then
  begin
    Memo1.Lines.LoadFromFile(OpenDialog1.FileName);

    Printer.BeginDoc;
    X := 200;
    Y := 200;
    for I := 0 to 140 do
      if (not Printer.Aborted) then
      begin
        Printer.Canvas.TextOut(X, Y, Memo1.Lines[I]);
        Y := Y + 80;
        if (Y > (Printer.PageHeight - 300)) then
        begin
          Y := 200;
          Printer.NewPage;
          Sleep(1000);  // to give you time to abort!
        end;
      end;
    if (not Printer.Aborted) then Printer.EndDoc;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
   KeyPreview := True;
end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
if (Key=VK_ESCAPE) and Printer.Printing then
  begin
  Printer.Abort;
  MessageDlg('Printing aborted', mtInformation, [mbOK],0);
  end;
end;
```

See Also

Abort (see page 3041)

BeginDoc (see page 3043)

EndDoc (see page 3045)

57.1.2.5.14 TPrinter.Title Property

Determines the text that is listed in the Print Manager and on network header pages.

Pascal

```
property Title: string;
```

C++

```
__property UnicodeString Title;
```

Description

Title is used by Windows in the Print Manager and for network title pages.

57.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	AssignPrn (see page 3066)	Assigns a text file variable to the printer.
	Printer (see page 3066)	Returns a global instance of TPrinter (see page 3038) to manage interaction with the printer.
	SetPrinter (see page 3068)	Replaces the global instance of TPrinter (see page 3038) that manages interaction with the printer.

57.2.1 Printers.AssignPrn Function

Assigns a text file variable to the printer.

Pascal

```
procedure AssignPrn(var F: Text);
```

C++

```
AssignPrn(Text F);
```

File

Printers

Description

In Delphi code, Call AssignPrn to cause all information written to the text file variable F to be directed to the printer. After the variable is assigned, the application must call the Rewrite procedure.

See Also

Rewrite

57.2.2 Printers.Printer Function

Returns a global instance of TPrinter (see page 3038) to manage interaction with the printer.

Pascal

```
function Printer: TPrinter;
```

C++

```
TPrinter Printer();
```

File

Printers

Description

Printer creates an instance of a global TPrinter (see page 3038) object the first time it is called. Use the Printer function to print

using the TPrinter (see page 3038) object.

Printer is declared in the Printers (see page 3037) unit. To use the Printer function and the TPrinter (see page 3038) object, add Printers (see page 3037) to the uses clause of your unit (Delphi) or include Printers.hpp in the source file (C++).

C++ Examples:

```
{
This example uses a button, a Page Control, and a Print
dialog box on a form. When the user clicks the button, the
print dialog is displayed.  The user can select any subset
of the pages in the page control for printing.  The selected
pages are then printed.  To run this example successfully,
you must add the Printers unit to the uses clause of your
unit.  Right click on the PageControl to add at least one
page to the PageControl.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  PrintDialog1->Options.Clear();
  PrintDialog1->Options << poPageNums << poSelection;
  PrintDialog1->FromPage = 1;
  PrintDialog1->MinPage = 1;
  PrintDialog1->ToPage = PageControl1->PageCount;
  PrintDialog1->MaxPage = PageControl1->PageCount;
  if (PrintDialog1->Execute())
  {
    int Start, Stop;
    // determine the range the user wants to print
    switch (PrintDialog1->PrintRange)
    {
      case prSelection:
        Start = PageControl1->ActivePage->PageIndex;
        Stop = Start;
        break;
      case prPageNums:
        Start = PrintDialog1->FromPage - 1;
        Stop =  PrintDialog1->ToPage - 1;
        break;
      default:  // prAllPages
        Start = PrintDialog1->MinPage - 1;
        Stop = PrintDialog1->MaxPage - 1;
        break;
    }
    // now, print the pages
    Printer()->BeginDoc();
    for (int i = Start; i <= Stop; i++)
    {
      PageControl1->Pages[i]->PaintTo(Printer()->Handle, 10, 10);
      if (i != Stop)
        Printer()->NewPage();
    }
    Printer()->EndDoc();
  }
}
```

Delphi Examples:

```
{
This example uses a button, a Page Control, and a Print
dialog box on a form. When the user clicks the button, the
print dialog is displayed.  The user can select any subset
of the pages in the page control for printing.  The selected
pages are then printed.  To run this example successfully,
you must add the Printers unit to the uses clause of your
unit.  Right click on the PageControl to add at least one
page to the PageControl.
```

```
}
procedure TForm1.Button1Click(Sender:TObject);
var
  I, Start, Stop: Integer;
begin
PrintDialog1.Options := [poPageNums, poSelection];
PrintDialog1.FromPage := 1;
PrintDialog1.MinPage := 1;
PrintDialog1.ToPage := PageControl1.PageCount;
PrintDialog1.MaxPage := PageControl1.PageCount;

if PrintDialog1.Execute() then
  begin
    { determine the range the user wants to print }
    with PrintDialog1 do
    begin
      if PrintRange = prAllPages then
        begin
        Start := MinPage - 1;
        Stop := MaxPage - 1;
      end
      else if PrintRange = prSelection then
      begin
        Start := PageControl1.ActivePage.PageIndex;
        Stop := Start;
        end
      else  { PrintRange = prPageNums }
      begin
        Start := FromPage - 1;
        Stop := ToPage - 1;
      end;
    end;
    { now, print the pages }
    with Printer do
    begin
      BeginDoc;
      for I := Start to Stop do
      begin
        PageControl1.Pages[I].PaintTo(Handle, 10, 10);
        if I <> Stop then
          NewPage;
      end;
      EndDoc;
    end;
  end;
end;
```

See Also

SetPrinter (see page 3068)

57.2.3 Printers.SetPrinter Function

Replaces the global instance of TPrinter (see page 3038) that manages interaction with the printer.

Pascal

```
function SetPrinter(NewPrinter: TPrinter): TPrinter;
```

C++

```
TPrinter SetPrinter(TPrinter NewPrinter);
```

File

Printers

Description

SetPrinter replaces the global TPrinter (see page 3038) object with another TPrinter (see page 3038) object. This allows applications to change the way they handle printing by substituting a TPrinter (see page 3038) descendant for the default TPrinter (see page 3038) object.

NewPrinter is the new TPrinter (see page 3038) object that replaces the current global TPrinter (see page 3038) object.

SetPrinter returns the previous global TPrinter (see page 3038) object.

Note: The global TPrinter (see page 3038) object is freed automatically when the application shuts down. After a call to SetPrinter, the printer that is returned is not automatically freed. It is the caller's responsibility to either free the return value, or replace it using another call to SetPrinter and to free the substitute printer that the second SetPrinter call returns.

SetPrinter is declared in the Printers (see page 3037) unit. To use the SetPrinter function, add Printers (see page 3037) to the uses clause of your unit (Delphi) or include Printers.hpp in the source file (C++) .

See Also

Printer (see page 3066)

57.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TPrinterCapability (see page 3070)	TPrinterCapability and TPrinterCapabilities (see page 3072) specify the capabilities of a printer device driver.
	TPrinterOrientation (see page 3070)	TPrinterOrientation indicates the orientation of pages when printing.
	TPrinterState (see page 3071)	This is record Printers.TPrinterState.

57.3.1 Printers.TPrinterCapability Enumeration

TPrinterCapability and TPrinterCapabilities (see page 3072) specify the capabilities of a printer device driver.

Pascal

```
TPrinterCapability = (
  pcCopies,
  pcOrientation,
  pcCollation
);
```

C++

```
enum TPrinterCapability {
  pcCopies,
  pcOrientation,
  pcCollation
};
```

File

Printers

Description

TPrinterCapabilities (see page 3072) is a set of TPrinterCapability values. The possible values of TPrinterCapability are

Value	Meaning
pcCopies	The printer driver allows the user to specify the number of copies.
pcOrientation	The printer driver allows the user to specify portrait or landscape orientation.
pcCollation	The printer driver allows the user to specify whether copies should be collated.

57.3.2 Printers.TPrinterOrientation Enumeration

TPrinterOrientation indicates the orientation of pages when printing.

Pascal

```
TPrinterOrientation = (
```

```
    poPortrait,
    poLandscape
  );
```

C++

```
  enum TPrinterOrientation {
    poPortrait,
    poLandscape
  };
```

File

Printers

Description

These are the possible values for TPrinterOrientation:

Value	Meaning
poPortrait	The job prints vertically on the page.
poLandscape	The job prints horizontally on the page.

57.3.3 Printers.TPrinterState Enumeration

Pascal

```
  TPrinterState = (
    psNoHandle,
    psHandleIC,
    psHandleDC
  );
```

C++

```
  enum TPrinterState {
    psNoHandle,
    psHandleIC,
    psHandleDC
  };
```

File

Printers

Description

This is record Printers.TPrinterState.

57.4 Types

The following table lists types in this documentation.

Types

Name	Description
TPrinterCapabilities (see page 3072)	TPrinterCapabilities and TPrinterCapability (see page 3070) specify the capabilities of a printer device driver.

57.4.1 Printers.TPrinterCapabilities Type

TPrinterCapabilities and TPrinterCapability (see page 3070) specify the capabilities of a printer device driver.

Pascal

```
TPrinterCapabilities = set of TPrinterCapability;
```

C++

```
TPrinterCapability TPrinterCapabilities;
```

File

Printers

Description

TPrinterCapabilities is a set of TPrinterCapability (see page 3070) values. The following table lists the possible values of TPrinterCapability (see page 3070).

Value	Meaning
pcCopies	The printer driver allows the user to specify the number of copies.
pcOrientation	The printer driver allows the user to specify portrait or landscape orientation.
pcCollation	The printer driver allows the user to specify whether copies should be collated.

See Also

TPrinterCapability (see page 3070)

58 Provider Namespace

This is namespace Provider.

Topics

Name	Description
Classes (see page 3075)	The following table lists classes in this documentation.
Interfaces (see page 3213)	The following table lists interfaces in this documentation.
Functions (see page 3215)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 3217)	The following table lists structs, records, enums in this documentation.
Types (see page 3222)	The following table lists types in this documentation.
Variables (see page 3231)	The following table lists variables in this documentation.
Constants (see page 3232)	The following table lists constants in this documentation.

Classes

	Name	Description
	EDSWriter (see page 3075)	EDSWriter is the exception class for errors that occur when creating data packets to represent the data or metadata in a dataset.
	TBaseProvider (see page 3077)	TBaseProvider is a base class for all providers that represent the data in a dataset.
	TCustomPacketWriter (see page 3093)	This is class Provider.TCustomPacketWriter.
	TCustomProvider (see page 3096)	TCustomProvider is the base class for data providers.
	TCustomResolver (see page 3125)	TCustomResolver is the base class for components that apply updates on behalf of a provider.
	TDataPacketWriter (see page 3136)	This is class Provider.TDataPacketWriter.
	TDataSetProvider (see page 3145)	TDataSetProvider provides data from a dataset and resolves updates to that dataset or its database server.
	TDataSetResolver (see page 3176)	TDataSetResolver applies updates to a dataset on behalf of a provider.
	TLocalAppServer (see page 3182)	This is class Provider.TLocalAppServer.
	TPacketDataSet (see page 3188)	TPacketDataSet represents a data packet or delta packet that is used by a provider.
	TProvider (see page 3196)	TProvider provides data to and applies updates from a client dataset.
	TSQLResolver (see page 3196)	TSQLResolver applies updates to a database server on behalf of a provider.
	TUpdateTree (see page 3205)	TUpdateTree stores information about a delta packet on behalf of a resolver.

Constants

Name	Description
MetaDataOption (see page 3232)	Specifies the options available when retrieving metadata from a target database.

ResetOption (see page 3233)	This is constant Provider.ResetOption.
XMLOption (see page 3233)	This is constant Provider.XMLOption.
XMLUTF8Option (see page 3234)	This is constant Provider.XMLUTF8Option.

Functions

	Name	Description
	GetObjectProperty (see page 3215)	This is function Provider.GetObjectProperty.
	GetStringProperty (see page 3215)	This is function Provider.GetStringProperty.
	VarArrayFromStrings (see page 3215)	Creates a Variant array that contains the strings in a TStrings object.
	VarArrayFromWideStrings (see page 3216)	This is function Provider.VarArrayFromWideStrings.

Interfaces

	Name	Description
	IProviderContainer (see page 3213)	This is class Provider.IProviderContainer.

Structs, Records, Enums

	Name	Description
	TGetRecordOption (see page 3217)	TGetRecordOption enumerates the types of information that can be added to data packets.
	TProviderOption (see page 3218)	TProviderOption and TProviderOptions (see page 3228) influence the type of information that is communicated between client datasets and provider components.
	TPutFieldInfo (see page 3220)	TPutFieldInfo contains information about a field.
	TResolverResponse (see page 3221)	TResolverResponse indicates the action a resolver component should take after an OnUpdateError event handler handles an update error.

Types

Name	Description
PPutFieldInfo (see page 3222)	PPutFieldInfo is a pointer type to a TPutFieldInfo (see page 3220) record.
TAfterUpdateRecordEvent (see page 3223)	TAfterUpdateRecordEvent represents the type of events that occur when the associated resolver has just updated a record.
TBeforeCommitEvent (see page 3223)	TBeforeCommitEvent represents the procedure type for the events that occur before committing a data set.
TBeforeUpdateRecordEvent (see page 3224)	TBeforeUpdateRecordEvent represents the procedure type for the events that occur before a record is updated.
TDataRequestEvent (see page 3224)	Represents the function type for the events that occur when a provider calls its DataRequest method.
TGetDSProps (see page 3225)	Represents the procedure type for the events that occur when the provider adds additional information to a data packet it is creating.
TGetParamsEvent (see page 3225)	Represents the procedure type for the events that occur when the provider adds attributes to the data set.
TGetRecordOptions (see page 3226)	TGetRecordOptions is used to decide what type of information is added to data packets.
TGetTableNameEvent (see page 3226)	Represents the procedure type for the events that occur when a resolver initializes its information about the table to which it applies updates.
TInfoArray (see page 3227)	TInfoArray is an array of TPutFieldInfo (see page 3220) records.
TOnValidate (see page 3227)	Represents the procedure type for the events that occur when the provider applies updates contained in a `Delta` packet.
TProviderDataEvent (see page 3228)	Represents the procedure type for the events that occur when the provider applies updates or retrieves data.
TProviderOptions (see page 3228)	TProviderOptions influences the type of information that is communicated between client datasets and provider components.
TPutFieldProc (see page 3229)	Allows to decide the procedure used when adding a new field to a data set.
TResolverErrorEvent (see page 3230)	Represents the procedure type for the events that occur when the provider fails to update a record.

Variables

Name	Description
InformixLob (see page 3231)	Resolves Informix rather than Oracle BLOB and CLOB fields.

58.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	EDSWriter (see page 3075)	EDSWriter is the exception class for errors that occur when creating data packets to represent the data or metadata in a dataset.
	TBaseProvider (see page 3077)	TBaseProvider is a base class for all providers that represent the data in a dataset.
	TCustomPacketWriter (see page 3093)	This is class Provider.TCustomPacketWriter.
	TCustomProvider (see page 3096)	TCustomProvider is the base class for data providers.
	TCustomResolver (see page 3125)	TCustomResolver is the base class for components that apply updates on behalf of a provider.
	TDataPacketWriter (see page 3136)	This is class Provider.TDataPacketWriter.
	TDataSetProvider (see page 3145)	TDataSetProvider provides data from a dataset and resolves updates to that dataset or its database server.
	TDataSetResolver (see page 3176)	TDataSetResolver applies updates to a dataset on behalf of a provider.
	TLocalAppServer (see page 3182)	This is class Provider.TLocalAppServer.
	TPacketDataSet (see page 3188)	TPacketDataSet represents a data packet or delta packet that is used by a provider.
	TProvider (see page 3196)	TProvider provides data to and applies updates from a client dataset.
	TSQLResolver (see page 3196)	TSQLResolver applies updates to a database server on behalf of a provider.
	TUpdateTree (see page 3205)	TUpdateTree stores information about a delta packet on behalf of a resolver.

58.1.1 EDSWriter Class

EDSWriter is the exception class for errors that occur when creating data packets to represent the data or metadata in a dataset.

Class Hierarchy

Pascal

```
EDSWriter = class(Exception);
```

C++

```
class EDSWriter : public Exception;
```

File

Provider

Description

EDSWriter is raised whenever a provider component encounters an error while trying to create a data packet from a dataset.

See Also

TDataSetProvider (see page 3145)

TCustomProvider (see page 3096)

58.1.1.1 EDSWriter Members

The following tables list the members exposed by EDSWriter.

Public Methods

	Name	Description
	Create (see page 3076)	Creates an instance of EDSWriter (see page 3075) using a specified error code and message string.

Public Properties

	Name	Description
	ErrorCode (see page 3076)	Indicates the error code that this exception represents.

58.1.1.2 EDSWriter.Create Constructor

Creates an instance of EDSWriter (see page 3075) using a specified error code and message string.

Pascal

```
constructor Create(ErrMsg: string; Status: Integer);
```

C++

```
__fastcall EDSWriter(UnicodeString ErrMsg, int Status);
```

Description

Create instantiates EDSWriter (see page 3075), setting the Message property to ErrMsg and the ErrorCode (see page 3076) property to Status.

See Also

Create

ErrorCode (see page 3076)

58.1.1.3 EDSWriter Properties

The properties of the EDSWriter class are listed here.

Public Properties

	Name	Description
	ErrorCode (see page 3076)	Indicates the error code that this exception represents.

58.1.1.3.1 EDSWriter.ErrorCode Property

Indicates the error code that this exception represents.

Pascal

```
property ErrorCode: Integer;
```

C++

```
__property int ErrorCode;
```

Description

Use ErrorCode to determine what error gave rise to this exception. ErrorCode is the value passed to the canonical EDSWriter (

see page 3075) constructor (C++) or Create (see page 3076) method (Delphi).

See Also

Create (see page 3076)

58.1.2 TBaseProvider Class

TBaseProvider is a base class for all providers that represent the data in a dataset.

Class Hierarchy

Pascal

```
TBaseProvider = class(TCustomProvider);
```

C++

```
class TBaseProvider : public TCustomProvider;
```

File

Provider

Description

TBaseProvider is the immediate ancestor of TDataSetProvider (see page 3145), which provides data from any dataset and applies updates back to that dataset or directly to an SQL server. Use TBaseProvider to derive other customized data provider objects, such as providers that apply updates directly to a database without using SQL.

Unlike its immediate ancestor (TCustomProvider (see page 3096)) TBaseProvider includes no abstract (Delphi) or pure virtual (C++) methods. However, applications should not create instances of TBaseProvider, as many of the methods are merely stubs that provide a backstop for specialized descendants that do not implement all the provider methods. To add a provider component to a form or data module, use TDataSetProvider (see page 3145) (or TXMLTransformProvider).

A provider derived from TBaseProvider is an intermediate link between a TClientDataSet or TXMLBroker object and a source dataset. The source dataset can reside in the same application as the client dataset or XML broker, or it can be part of a separate application server. The provider packages data from the source dataset and passes it in one or more transportable data packets to the client dataset or XML broker. The client dataset reconstructs the data in the data packet to create a local, in-memory copy for user access. When user access is complete, the client dataset repackages any changed data and sends the updates back to the provider. The provider applies the updates to the database or source dataset.

When the provider supplies data to a client dataset, the client dataset reconstructs the data in the data packet to create a local, in-memory copy for user access. When user access is complete, the client dataset repackages any changed data and sends the updates back to the provider. The provider applies the updates to the database or source dataset.

When the provider supplies data to an XML broker, the XML broker adds the data packet, in XML format, to an HTML document that is downloaded by a Web client. When the XML broker receives updates from the Web client, it sends the updates to the provider, which applies them to the database or source dataset.

Client datasets and XML brokers communicate with a provider through the IAppServer interface. In multi-tiered applications, this is the interface of the remote data module that contains the provider. To enable the remote data module to pass calls to the provider, the provider's Exported (see page 3115) property must be true, and its Owner property must specify the remote data module.

See Also

TClientDataSet

TDataSetProvider (see page 3145)

IAppServer

TCustomProvider (see page 3096)

TXMLTransformProvider

TSoapDataModule

58.1.2.1 TBaseProvider Members

The following tables list the members exposed by TBaseProvider.

Protected Events

	Name	Description
	AfterApplyUpdates (see page 3116)	Occurs after the provider applies updates.
	AfterExecute (see page 3117)	Occurs after the provider passes an execute command to its dataset.
	AfterGetParams (see page 3118)	Occurs after the provider creates an OleVariant containing parameter values to send to the client dataset.
	AfterGetRecords (see page 3118)	Occurs after the provider creates a data packet to send to the client dataset.
	AfterRowRequest (see page 3119)	Occurs after the provider fetches new information about the current record.
	BeforeApplyUpdates (see page 3120)	Occurs before the provider applies updates from a client dataset.
	BeforeExecute (see page 3121)	Occurs before the provider passes an execute command to its dataset.
	BeforeGetParams (see page 3121)	Occurs before the provider creates an OleVariant that encodes the current parameter values and sends them to a client dataset.
	BeforeGetRecords (see page 3122)	Occurs before the provider creates a data packet and sends it to a client dataset.
	BeforeRowRequest (see page 3123)	Occurs before the provider creates a delta packet with information about the current record.
	OnValidate (see page 3125)	This is OnValidate, a member of class TCustomProvider.

Protected Methods

	Name	Description
	CheckResolver (see page 3081)	Instantiates a resolver for the Resolver (see page 3088) property if it does not already exist.
	CreateDataPacket (see page 3081)	Generates a data packet for the data associated with the provider.
	CreateResolver (see page 3082)	Creates a resolver component for handling the application of updates.
	DoAfterApplyUpdates (see page 3102)	Generates an AfterApplyUpdates (see page 3116) event.
	DoAfterExecute (see page 3102)	Generates an AfterExecute (see page 3117) event.
	DoAfterGetParams (see page 3102)	Generates an AfterGetParams (see page 3118) event.
	DoAfterGetRecords (see page 3103)	Generates an AfterGetRecords (see page 3118) event.
	DoAfterRowRequest (see page 3103)	Generates an AfterRowRequest (see page 3119) event.
	DoAfterUpdateRecord (see page 3082)	Generates an AfterUpdateRecord (see page 3090) event.
	DoApplyUpdates (see page 3104)	This is DoApplyUpdates, a member of class TCustomProvider.
	DoBeforeApplyUpdates (see page 3104)	Generates a BeforeApplyUpdates (see page 3120) event.
	DoBeforeCommit (see page 3082)	This is DoBeforeCommit, a member of class TBaseProvider.
	DoBeforeExecute (see page 3105)	Generates a BeforeExecute (see page 3121) event.
	DoBeforeGetParams (see page 3105)	Generates a BeforeGetParams (see page 3121) event.
	DoBeforeGetRecords (see page 3106)	Generates a BeforeGetRecords (see page 3122) event.
	DoBeforeRowRequest (see page 3106)	Generates a BeforeRowRequest (see page 3123) event.
	DoBeforeUpdateRecord (see page 3083)	This is DoBeforeUpdateRecord, a member of class TBaseProvider.
	DoGetRecords (see page 3107)	This is DoGetRecords, a member of class TCustomProvider.
	DoOnGetData (see page 3083)	Generates an OnGetData (see page 3091) event.
	DoOnUpdateData (see page 3083)	Generates an OnUpdateData (see page 3092) event.
	FetchDetails (see page 3083)	Fills a delta dataset with all nested dataset field values.
	FreeResolver (see page 3084)	Frees the resolver specified by the Resolver (see page 3088) property.

	InternalApplyUpdates (see page 3084)	Provides the underlying implementation of the ApplyUpdates (see page 3100) method.
	InternalExecute (see page 3111)	Provides the underlying implementation of the Execute (see page 3107) method.
	InternalGetParams (see page 3111)	Returns the requested parameter values as a Variant array.
	InternalGetRecords (see page 3085)	Provides the underlying implementation of the GetRecords (see page 3109) method.
	InternalRowRequest (see page 3086)	Provides the underlying implementation of the RowRequest (see page 3113) method.
	LocateRecord (see page 3086)	Positions a data set on the record that corresponds to the current update in a specified delta packet.
	SetActiveUpdateException (see page 3114)	This is SetActiveUpdateException, a member of class TCustomProvider.
	UpdateRecord (see page 3087)	Retrieves specified data from a source dataset into a delta packet.

Protected Properties

	Name	Description
	ActiveUpdateException (see page 3114)	This is ActiveUpdateException, a member of class TCustomProvider.
	OwnerData (see page 3115)	This is OwnerData, a member of class TCustomProvider.
	PacketDataSet (see page 3088)	This is PacketDataSet, a member of class TBaseProvider.

Public Events

	Name	Description
	AfterUpdateRecord (see page 3090)	Occurs when the associated resolver has just updated a record.
	BeforeUpdateRecord (see page 3090)	Occurs immediately before each record is applied to the remote dataset.
	OnDataRequest (see page 3091)	Occurs when a provider calls its DataRequest method.
	OnGetData (see page 3091)	Occurs after a provider retrieves data but before it is sent to the client.
	OnUpdateData (see page 3092)	Occurs when a provider begins to apply updates.
	OnUpdateError (see page 3092)	Occurs when a provider can't update a record because of an error condition.

Public Methods

	Name	Description
	ApplyUpdates (see page 3100)	This is the overview for the ApplyUpdates method overload.
	Create (see page 3079)	Creates and initializes a TBaseProvider (see page 3077) object.
	DataRequest (see page 3101)	Generates an OnDataRequest (see page 3124) event.
	Destroy (see page 3080)	Frees an instance of a provider object.
	Execute (see page 3107)	Executes a query, stored procedure, or custom SQL statement.
	GetParams (see page 3108)	Returns the current parameter values.
	GetRecords (see page 3109)	This is the overview for the GetRecords method overload.
	RowRequest (see page 3113)	Returns information from a specified record.

Public Properties

	Name	Description
	Data (see page 3115)	Denotes a data packet created by the provider.
	Exported (see page 3115)	Allows client applications to pass calls to the provider when it resides on an application server.
	Options (see page 3087)	Influences what is included in data packets and how the information in data packets is used.
	Resolver (see page 3088)	Provides access to the resolver component that applies updates and resolves update errors.
	UpdateMode (see page 3089)	Controls how the provider locates records when applying updates.

58.1.2.2 TBaseProvider.Create Constructor

Creates and initializes a TBaseProvider (see page 3077) object.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TBaseProvider(TComponent * AOwner);
```

Description

Call Create to instantiate a provider at runtime. Providers placed in forms or data modules at design time are created automatically.

AOwner specifies the component, typically a form or data module, that is responsible for freeing the provider.

See Also

Owner

58.1.2.3 TBaseProvider.Destroy Destructor

Frees an instance of a provider object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TBaseProvider();
```

Description

Do not call Destroy directly. Instead, call Free, which checks that the provider is not nil before calling the Destroy method.

See Also

System::TObject::Free

Resolver (see page 3088)

58.1.2.4 TBaseProvider Methods

The methods of the TBaseProvider class are listed here.

Protected Methods

	Name	Description
	CheckResolver (see page 3081)	Instantiates a resolver for the Resolver (see page 3088) property if it does not already exist.
V	CreateDataPacket (see page 3081)	Generates a data packet for the data associated with the provider.
V	CreateResolver (see page 3082)	Creates a resolver component for handling the application of updates.
V	DoAfterUpdateRecord (see page 3082)	Generates an AfterUpdateRecord (see page 3090) event.
V	DoBeforeCommit (see page 3082)	This is DoBeforeCommit, a member of class TBaseProvider.
V	DoBeforeUpdateRecord (see page 3083)	This is DoBeforeUpdateRecord, a member of class TBaseProvider.
V	DoOnGetData (see page 3083)	Generates an OnGetData (see page 3091) event.
V	DoOnUpdateData (see page 3083)	Generates an OnUpdateData (see page 3092) event.
V	FetchDetails (see page 3083)	Fills a delta dataset with all nested dataset field values.
	FreeResolver (see page 3084)	Frees the resolver specified by the Resolver (see page 3088) property.
V	InternalApplyUpdates (see page 3084)	Provides the underlying implementation of the ApplyUpdates (see page 3100) method.

	InternalGetRecords (see page 3085)	Provides the underlying implementation of the GetRecords (see page 3109) method.
	InternalRowRequest (see page 3086)	Provides the underlying implementation of the RowRequest (see page 3113) method.
	LocateRecord (see page 3086)	Positions a data set on the record that corresponds to the current update in a specified delta packet.
	UpdateRecord (see page 3087)	Retrieves specified data from a source dataset into a delta packet.

58.1.2.4.1 TBaseProvider.CheckResolver Method

Instantiates a resolver for the Resolver (see page 3088) property if it does not already exist.

Pascal

```
procedure CheckResolver;
```

C++

```
__fastcall CheckResolver();
```

Description

CheckResolver is called internally when the provider needs to use its resolver component. It checks whether Resolver (see page 3088) is nil (Delphi) or NULL (C++), and if so, calls CreateResolver (see page 3082) to assign a value.

See Also

Resolver (see page 3088)

CreateResolver (see page 3082)

58.1.2.4.2 TBaseProvider.CreateDataPacket Method

Generates a data packet for the data associated with the provider.

Pascal

```
procedure CreateDataPacket(PacketOpts: TGetRecordOptions; ProvOpts: TProviderOptions; var
RecsOut: Integer; var Data: OleVariant); virtual;
```

C++

```
virtual __fastcall CreateDataPacket(TGetRecordOptions PacketOpts, TProviderOptions ProvOpts,
int RecsOut, OleVariant Data);
```

Description

CreateDataPacket is used internally to generate data packets. As implemented in TBaseProvider (see page 3077), CreateDataPacket sets RecsOut to 0 and Data (see page 3115) to NULL. Descendants override this method to create the data packets that are returned by GetRecords (see page 3109).

The PacketOpts parameter indicates what information (other than record values) is included in the data packet. It reflects the Options (see page 3087) passed to the GetRecords (see page 3109) method.

The ProvOpts parameter reflects the current value of the Options (see page 3087) property.

The RecsOut parameter specifies the number of records desired in the packet and returns the actual number of records in the generated data packet.

The Data (see page 3115) parameter returns the generated data packet.

See Also

GetRecords (see page 3109)

Options (see page 3087)

58.1.2.4.3 TBaseProvider.CreateResolver Method

Creates a resolver component for handling the application of updates.

Pascal

```
function CreateResolver: TCustomResolver; virtual;
```

C++

```
virtual __fastcall TCustomResolver CreateResolver();
```

Description

CreateResolver is called automatically when a resolver is needed to implement one of TBaseProvider (see page 3077)'s methods. As implemented in TBaseProvider (see page 3077), CreateResolver always returns nil (Delphi) or NULL (C++). Descendants of TBaseProvider (see page 3077) override this method to create a TCustomResolver (see page 3125) descendant that applies updates in the desired manner.

See Also

Resolver (see page 3088)

FreeResolver (see page 3084)

58.1.2.4.4 TBaseProvider.DoAfterUpdateRecord Method

Generates an AfterUpdateRecord (see page 3090) event.

Pascal

```
procedure DoAfterUpdateRecord(SourceDS: TDataSet; DeltaDS: TCustomClientDataSet; UpdateKind:
TUpdateKind); virtual;
```

C++

```
virtual __fastcall DoAfterUpdateRecord(TDataSet SourceDS, TCustomClientDataSet DeltaDS,
TUpdateKind UpdateKind);
```

Description

DoAfterUpdateRecord is the protected implementation of the AfterUpdateRecord (see page 3090) event. Override this method to perform class-specific actions in response to data requests.

See Also

AfterUpdateRecord (see page 3090)

58.1.2.4.5 TBaseProvider.DoBeforeCommit Method

Pascal

```
procedure DoBeforeCommit(Delta: TPacketDataSet; ErrorCount: Integer; MaxErrors: Integer; const
ReturnVar: OleVariant); virtual;
```

C++

```
virtual __fastcall DoBeforeCommit(TPacketDataSet Delta, int ErrorCount, int MaxErrors, const
OleVariant ReturnVar);
```

Description

This is DoBeforeCommit, a member of class TBaseProvider.

58.1.2.4.6 TBaseProvider.DoBeforeUpdateRecord Method

Pascal

```
procedure DoBeforeUpdateRecord(SourceDS: TDataSet; DeltaDS: TCustomClientDataSet; UpdateKind:
TUpdateKind; var Applied: Boolean); virtual;
```

C++

```
virtual __fastcall DoBeforeUpdateRecord(TDataSet SourceDS, TCustomClientDataSet DeltaDS,
TUpdateKind UpdateKind, Boolean Applied);
```

Description

This is DoBeforeUpdateRecord, a member of class TBaseProvider.

58.1.2.4.7 TBaseProvider.DoOnGetData Method

Generates an OnGetData (see page 3091) event.

Pascal

```
procedure DoOnGetData(var Data: OleVariant); virtual;
```

C++

```
virtual __fastcall DoOnGetData(OleVariant Data);
```

Description

DoOnGetData is the protected implementation of the OnGetData (see page 3091) event. Override this method to perform class-specific actions in response to data requests.

See Also

OnGetData (see page 3091)

58.1.2.4.8 TBaseProvider.DoOnUpdateData Method

Generates an OnUpdateData (see page 3092) event.

Pascal

```
procedure DoOnUpdateData(Delta: TPacketDataSet); virtual;
```

C++

```
virtual __fastcall DoOnUpdateData(TPacketDataSet Delta);
```

Description

DoOnUpdateData is the protected implementation of the OnUpdateData (see page 3092) event. The Delta property contains the delta packet of records to update. It is passed to the OnUpdateData (see page 3092) event handler as a client dataset.

Override this method to perform class-specific actions in response to update requests.

See Also

OnUpdateData (see page 3092)

58.1.2.4.9 TBaseProvider.FetchDetails Method

Fills a delta dataset with all nested dataset field values.

Pascal

```
procedure FetchDetails(Source: TDataSet; Delta: TDataSet); virtual;
```

C++

```
virtual __fastcall FetchDetails(TDataSet Source, TDataSet Delta);
```

Description

FetchDetails is used internally to fetch the current values of any nested datasets when the provider is executing its RowRequest (see page 3113) method. As implemented in TBaseProvider (see page 3077), FetchDetails does nothing. Descendants override this method to add nested detail datasets to the delta packet specified by Delta.

Source is a source dataset that has the current values for all fields. Delta is a delta dataset that must be updated so that its nested detail fields match the values in Source.

See Also

RowRequest (see page 3113)

58.1.2.4.10 TBaseProvider.FreeResolver Method

Frees the resolver specified by the Resolver (see page 3088) property.

Pascal

```
procedure FreeResolver;
```

C++

```
__fastcall FreeResolver();
```

Description

Call FreeResolver to safely dispose of the component specified by the Resolver (see page 3088) property. FreeResolver sets the Resolver (see page 3088) property to nil (Delphi) or NULL (C++) after calling the resolver's Free method.

See Also

Resolver (see page 3088)

CreateResolver (see page 3082)

System::TObject::Free

58.1.2.4.11 TBaseProvider.InternalApplyUpdates Method

Provides the underlying implementation of the ApplyUpdates (see page 3100) method.

Pascal

```
function InternalApplyUpdates(const Delta: OleVariant; MaxErrors: Integer; out ErrorCount:
Integer): OleVariant; override;
```

C++

```
virtual __fastcall OleVariant InternalApplyUpdates(const OleVariant Delta, int MaxErrors, int
ErrorCount);
```

Description

The ApplyUpdates (see page 3100) method calls InternalApplyUpdates to apply the updates contained in a delta packet after generating the BeforeApplyUpdates (see page 3120) event and before generating the AfterApplyUpdates (see page 3116) event.

If the Options (see page 3087) property includes poReadOnly, InternalApplyUpdates raises an exception. Otherwise, it calls the ApplyUpdates (see page 3100) method of its Resolver (see page 3088) component to handle the actual update and resolving process. If the Resolver (see page 3088) property is nil (Delphi) or NULL (C++), ApplyUpdates (see page 3100) calls the CreateResolver (see page 3082) method and assigns the resulting resolver to the Resolver (see page 3088) property.

Delta is the delta packet containing insertions, deletions, and modifications to be applied.

MaxErrors indicates the maximum number of errors that can be applied before stopping the update operation. If MaxErrors is –1, the provider tries to apply all updates.

ErrorCount returns the number of errors encountered during the update operation.

InternalApplyUpdates returns a data packet of updates that could not be applied to the database as an OleVariant.

See Also

Resolver (see page 3088)

ApplyUpdates (see page 3128)

CreateResolver (see page 3082)

Options (see page 3087)

ApplyUpdates (see page 3100)

58.1.2.4.12 TBaseProvider.InternalGetRecords Method

Provides the underlying implementation of the GetRecords (see page 3109) method.

Pascal

```
function InternalGetRecords(Count: Integer; out RecsOut: Integer; Options: TGetRecordOptions;
const CommandText: WideString; var Params: OleVariant): OleVariant; override;
```

C++

```
virtual __fastcall OleVariant InternalGetRecords(int Count, int RecsOut, TGetRecordOptions
Options, const BSTR CommandText, OleVariant Params);
```

Description

The GetRecords (see page 3109) method calls InternalGetRecords to assemble a data packet after calling DoBeforeGetRecords (see page 3106) to apply the CommandText and Params and generate a BeforeGetRecords (see page 3122) event. InternalGetRecords calls CreateDataPacket (see page 3081) to create a data packet that reflects the current property settings of the provider. It then generates an OnGetData (see page 3091) event.

Count indicates the number of records to retrieve. If Count is -1, all records are retrieved. If Count is 0, only metadata is retrieved. If Count is greater than 0, only the specified number of records are retrieved.

RecsOut returns the actual number of records retrieved.

Options (see page 3087) indicates what information should be added to the data packet in addition to data. It can include grMetaData (include metadata), grReset (start providing records with the first one in the dataset), and grXML (encode the datapacket in XML rather than as an OleVariant) or grXMLUTF8 (same as grXML except extended characters are encoded using UTF8).

CommandText specifies an optional SQL statement that replaces the SQL of an associated query, or the name of a table or stored procedure that replaces the associated table or stored procedure. This parameter is ignored if the Options (see page 3087) property does not include poAllowCommandText.

Params specifies any parameter values that should be applied to CommandText or to an associated query or stored procedure.

If the provider is associated with a table, Params specifies field values that filter the result set. On exit from GetRecords (see page 3109), Params returns any output parameters.

Records are returned as a data packet in a Variant.

See Also

Data (see page 3115)

CreateDataPacket (see page 3081)

GetRecords (see page 3109)

OnGetData (see page 3091)

58.1.2.4.13 TBaseProvider.InternalRowRequest Method

Provides the underlying implementation of the RowRequest (see page 3113) method.

Pascal

```
function InternalRowRequest(const Row: OleVariant; RequestType: TFetchOptions): OleVariant;
override;
```

C++

```
virtual __fastcall OleVariant InternalRowRequest(const OleVariant Row, TFetchOptions
RequestType);
```

Description

The RowRequest (see page 3113) method calls InternalRowRequest to assemble the specified information after generating a BeforeRowRequest (see page 3123) event and before generating an AfterRowRequest (see page 3119) event. InternalRowRequest passes this request on to its associated Resolver (see page 3088).

The Row parameter is an OleVariant that describes the record for which information is requested. The provider uses Row to locate the record before assembling the requested information.

InternalRowRequest returns the requested information in a delta packet.

See Also

RowRequest (see page 3113)

Resolver (see page 3088)

RowRequest (see page 3134)

58.1.2.4.14 TBaseProvider.LocateRecord Method

Positions a data set on the record that corresponds to the current update in a specified delta packet.

Pascal

```
procedure LocateRecord(Source: TDataSet; Delta: TDataSet); virtual;
```

C++

```
virtual __fastcall LocateRecord(TDataSet Source, TDataSet Delta);
```

Description

LocateRecord is called indirectly by the RowRequest (see page 3113) method to locate a record before fetching information about a specified record. The Source parameter is the dataset containing current record values. The Delta parameter is a delta packet that identifies the desired type of information and information identifying the desired record.

As implemented in TBaseProvider (see page 3077), LocateRecord does nothing. Descendants override this method to locate the specified record in Source and navigate to it.

See Also

RowRequest (see page 3113)

58.1.2.4.15 TBaseProvider.UpdateRecord Method

Retrieves specified data from a source dataset into a delta packet.

Pascal

```
procedure UpdateRecord(Source: TDataSet; Delta: TDataSet; BlobsOnly: Boolean; KeyOnly:
Boolean); virtual;
```

C++

```
virtual __fastcall UpdateRecord(TDataSet Source, TDataSet Delta, Boolean BlobsOnly, Boolean
KeyOnly);
```

Description

UpdateRecord is called indirectly by the RowRequest (see page 3113) method to fetch BLOB fields or nested detail sets. The Source parameter is the dataset containing the BLOB fields or the detail dataset. The Delta parameter is a delta packet that returns the requested data. BlobsOnly indicates whether all field values should be provided (BlobsOnly == false) or only Blob field values (BlobsOnly == true).

As implemented in TBaseProvider (see page 3077), UpdateRecord does nothing. Descendants override this method to fill the Delta parameter with the requested information.

See Also

RowRequest (see page 3113)

58.1.2.5 TBaseProvider Properties

The properties of the TBaseProvider class are listed here.

Protected Properties

	Name	Description
	PacketDataSet (see page 3088)	This is PacketDataSet, a member of class TBaseProvider.

Public Properties

	Name	Description
	Options (see page 3087)	Influences what is included in data packets and how the information in data packets is used.
R	Resolver (see page 3088)	Provides access to the resolver component that applies updates and resolves update errors.
	UpdateMode (see page 3089)	Controls how the provider locates records when applying updates.

58.1.2.5.1 TBaseProvider.Options Property

Influences what is included in data packets and how the information in data packets is used.

Pascal

```
property Options: TProviderOptions;
```

C++

```
__property TProviderOptions Options;
```

Description

Set Options to customize how the provider communicates with client datasets. Options is a set that specifies

Whether nested detail sets and BLOB data are included in data packets or fetched separately.

Whether data packets include field component properties such as formats, display names, and maximum & minimum values.

Whether data packets are provided on a read-only basis, and whether specific types of changes (insertions, edits, or deletions) are disabled.

Whether changes to master table fields should cause corresponding updates to detail tables.

Whether a single update can affect multiple server records.

Whether the client's records are refreshed when it applies updates.

Whether the client can send SQL statements that override the provider's dataset.

Note: Not all options are supported by an InternetExpress client. Specifically, InternetExpress clients don't support poFetchBlobsOnDemand, poFetchDetailsOnDemand, poDisableInserts, poDisableEdits, psDisableDeletes, poPropogateChanges, and poAllowCommandText.

See Also

Data (see page 3115)

ApplyUpdates (see page 3100)

Overriding the Dataset On the Application Server

Setting Options That Influence the Data Packets

58.1.2.5.2 TBaseProvider.PacketDataSet Property

Pascal

```
property PacketDataSet: TPacketDataSet;
```

C++

```
__property TPacketDataSet PacketDataSet;
```

Description

This is PacketDataSet, a member of class TBaseProvider.

58.1.2.5.3 TBaseProvider.Resolver Property

Provides access to the resolver component that applies updates and resolves update errors.

Pascal

```
property Resolver: TCustomResolver;
```

C++

```
__property TCustomResolver Resolver;
```

Description

Provider (see page 3073) components use a resolver to handle the details of applying updates and resolving update errors. Most applications do not need to use the resolver directly.

When any method that uses a resolver is called, the provider automatically creates a resolver component (using the CreateResolver (see page 3082) method) if the value of Resolver is nil (Delphi) or NULL (C++). Thus, this property is nil (Delphi) or NULL (C++) until the first method that requires a resolver is called.

See Also

ApplyUpdates (see page 3100)

CreateResolver (see page 3082)

58.1.2.5.4 TBaseProvider.UpdateMode Property

Controls how the provider locates records when applying updates.

Pascal

```
property UpdateMode: TUpdateMode;
```

C++

```
__property TUpdateMode UpdateMode;
```

Description

Use UpdateMode to specify the criteria for locating a record in the dataset when applying updates. UpdateMode specifies whether modified records are located based on all columns (fields), on only the key fields, or on the key fields plus the original values of fields that have been modified.

Note: To control how the provider locates records by specifying individual fields, use the field components' ProviderFlags property.

See Also

ApplyUpdates (see page 3100)

ProviderFlags

Influencing How Updates Are Applied

58.1.2.6 TBaseProvider Events

The events of the TBaseProvider class are listed here.

Public Events

	Name	Description
	AfterUpdateRecord (see page 3090)	Occurs when the associated resolver has just updated a record.
	BeforeUpdateRecord (see page 3090)	Occurs immediately before each record is applied to the remote dataset.
	OnDataRequest (see page 3091)	Occurs when a provider calls its DataRequest method.
	OnGetData (see page 3091)	Occurs after a provider retrieves data but before it is sent to the client.
	OnUpdateData (see page 3092)	Occurs when a provider begins to apply updates.
	OnUpdateError (see page 3092)	Occurs when a provider can't update a record because of an error condition.

58.1.2.6.1 TBaseProvider.AfterUpdateRecord Event

Occurs when the associated resolver has just updated a record.

Pascal

```
property AfterUpdateRecord: TAfterUpdateRecordEvent;
```

C++

```
__property TAfterUpdateRecordEvent AfterUpdateRecord;
```

Description

Write an AfterUpdateRecord event handler to provide special processing immediately after a record has been successfully updated. For example, use AfterUpdateRecord to maintain a log of successful updates.

See Also

BeforeUpdateRecord (see page 3090)

AfterApplyUpdates (see page 3116)

58.1.2.6.2 TBaseProvider.BeforeUpdateRecord Event

Occurs immediately before each record is applied to the remote dataset.

Pascal

```
property BeforeUpdateRecord: TBeforeUpdateRecordEvent;
```

C++

```
__property TBeforeUpdateRecordEvent BeforeUpdateRecord;
```

Description

Write a BeforeUpdateRecord event handler to:

Prepare individual record updates before they are applied to the source dataset or the database server. For example, you can insert additional changes into a record before it is updated or set the ProviderFlags property of the fields in the DeltaDS parameter to indicate how the provider should identify the records that are to be updated or changed and which fields should be updated.

Manually apply updates that the provider can't handle. For example, the provider can't generate the appropriate SQL when the data results from a multi-table join and more than one table must be updated. Write code in the event handler to apply the update and then set the Applied parameter to true so that the provider does not generate an error.

Selectively prevent certain updates from occurring. By setting the Applied parameter to true, the default update does not occur. When you prevent updates in this way, the provider does not generate an update error.

See Also

AfterUpdateRecord (see page 3090)

ApplyUpdates (see page 3100)

OnUpdateData (see page 3092)

OnUpdateError (see page 3092)

ProviderFlags

BeforeApplyUpdates (see page 3120)

58.1.2.6.3 TBaseProvider.OnDataRequest Event

Occurs when a provider calls its DataRequest method.

Pascal

```
property OnDataRequest: TDataRequestEvent;
```

C++

```
__property TDataRequestEvent OnDataRequest;
```

Description

Write an OnDataRequest event handler to do custom processing of data. OnDataRequest has no built-in use. Instead, it allows developers to augment the IAppServer interface so that clients in a multi-tiered application can communicate directly with an individual provider.

If an application does not provide an OnDataRequest event handler, calls to DataRequest return NULL.

Note: To provide the same sort of custom processing on the client side, use an automation method on the connection component.

See Also

DataRequest (see page 3101)

IAppServer

Communicating with Providers Using Custom Events

Communicating with the Client Dataset

Responding to Client-generated Events

58.1.2.6.4 TBaseProvider.OnGetData Event

Occurs after a provider retrieves data but before it is sent to the client.

Pascal

```
property OnGetData: TProviderDataEvent;
```

C++

```
__property TProviderDataEvent OnGetData;
```

Description

Write an OnGetData event handler to perform any custom data processing the provider should handle before returning data to a client. The OnGetData event occurs after a provider fetches data from its dataset but before the data is sent on to the client. Using the methods of the DataSet parameter, data can be edited in the OnGetData event handler before sending it to the client dataset or XML broker.

Note: OnGetData occurs when the Data (see page 3115) property is read, as well as when the GetRecords (see page 3109) method is called.

See Also

GetRecords (see page 3109)

OnUpdateData (see page 3092)

BeforeGetRecords (see page 3122)

AfterGetRecords (see page 3118)

Responding to Client Data Requests

58.1.2.6.5 TBaseProvider.OnUpdateData Event

Occurs when a provider begins to apply updates.

Pascal

```
property OnUpdateData: TProviderDataEvent;
```

C++

```
__property TProviderDataEvent OnUpdateData;
```

Description

Write an OnUpdateData event to handle special processing before the provider begins applying updates received from the client dataset or XML broker. Using the methods belonging to the DataSet parameter, an application can

Examine data (for example, for values or data changes that should not permitted), and raise exceptions that cancel applying of updates before they occur.

Change data (for example encrypting or decrypting values) before it is sent on to the source dataset or database server.

See Also

ApplyUpdates (see page 3100)

BeforeUpdateRecord (see page 3090)

Adding Custom Information to Data Packets

Editing Delta Packets Before Updating the Database

Influencing How Updates Are Applied

58.1.2.6.6 TBaseProvider.OnUpdateError Event

Occurs when a provider can't update a record because of an error condition.

Pascal

```
property OnUpdateError: TResolverErrorEvent;
```

C++

```
__property TResolverErrorEvent OnUpdateError;
```

Description

Write an OnUpdateError event handler to respond to error conditions that arise when the provider applies a client dataset's updates. The OnUpdateError event handler is called for each record that can't be updated.

Warning: The provider can't detect when an update conflicts with another user's changes to a memo field. This situation will not generate an OnUpdateError event.

See Also

ApplyUpdates (see page 3100)

OnReconcileError

NewValue

OldValue

CurValue

Resolving Update Errors On the Provider

58.1.3 TCustomPacketWriter Class

Class Hierarchy

Pascal

```
TCustomPacketWriter = class(TObject);
```

C++

```
class TCustomPacketWriter : public TObject;
```

File

Provider

Description

This is class Provider.TCustomPacketWriter.

58.1.3.1 TCustomPacketWriter Members

The following tables list the members exposed by TCustomPacketWriter.

Protected Fields

	Name	Description
	FBuffer (see page 3095)	This is FBuffer, a member of class TCustomPacketWriter.

Protected Methods

	Name	Description
V	AddAttribute (see page 3095)	This is AddAttribute, a member of class TCustomPacketWriter.
	Check (see page 3095)	This is Check, a member of class TCustomPacketWriter.
	SetDSWriter (see page 3096)	This is SetDSWriter, a member of class TCustomPacketWriter.

Protected Properties

	Name	Description
R	DSWriter (see page 3096)	This is DSWriter, a member of class TCustomPacketWriter.

Public Methods

	Name	Description
V	Create (see page 3094)	Constructs an object and initializes its data before the object is first used.
V	Destroy (see page 3094)	Disposes of an object instance.

58.1.3.2 TCustomPacketWriter.Create Constructor

Constructs an object and initializes its data before the object is first used.

Pascal

```
constructor Create; virtual;
```

C++

```
virtual __fastcall TCustomPacketWriter();
```

Description

System::TObject::Create constructs an object. The purpose, size, and behavior of objects differ greatly. The System::TObject::Create constructor defined by System::TObject allocates memory but does not initialize data.

Descendant objects usually define a constructor that creates the particular kind of object and initializes its data.

Note: If an exception escapes from a constructor, the object's destructor is called to clean up the failed instance.

See Also

System::TObject::Destroy

System::TObject::Free

58.1.3.3 TCustomPacketWriter.Destroy Destructor

Disposes of an object instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomPacketWriter();
```

Description

Do not call System::TObject::Destroy directly. Call System::TObject::Free instead. Free verifies that the object reference is not nil before calling System::TObject::Destroy.

The System::TObject::Destroy method defined by System::TObject deallocates memory. Descendant objects usually define a destructor that is customized for that particular kind of object.

When declaring a System::TObject::Destroy method in a descendant, always add the override directive to the declaration and call the inherited System::TObject::Destroy as the last statement in the overriding method. Since System::TObject::Destroy is a virtual method, overriding it ensures that the proper inherited behavior occurs.

Note: If an exception escapes from the constructor, the destructor is called to destroy the partially constructed object instance that failed to initialize completely. Therefore, destructors should check that allocated resources, such as handles, were actually allocated before trying to release them, since their value might be zero.

System::TObject::Destroy should be implemented so that it calls System::TObject::Free on all subobjects created within the object's constructor (that is, allocated by the constructor). Unlike System::TObject::Destroy, System::TObject::Free provides a safeguard when destroying objects that are nil.

See Also

System::TObject::Free

58.1.3.4 TCustomPacketWriter Fields

The fields of the TCustomPacketWriter class are listed here.

Protected Fields

	Name	Description
	FBuffer (see page 3095)	This is FBuffer, a member of class TCustomPacketWriter.

58.1.3.4.1 TCustomPacketWriter.FBuffer Field

Pascal

```
FBuffer: TBlobByteData;
```

C++

```
TBlobByteData FBuffer;
```

Description

This is FBuffer, a member of class TCustomPacketWriter.

58.1.3.5 TCustomPacketWriter Methods

The methods of the TCustomPacketWriter class are listed here.

Protected Methods

	Name	Description
V	AddAttribute (see page 3095)	This is AddAttribute, a member of class TCustomPacketWriter.
	Check (see page 3095)	This is Check, a member of class TCustomPacketWriter.
	SetDSWriter (see page 3096)	This is SetDSWriter, a member of class TCustomPacketWriter.

58.1.3.5.1 TCustomPacketWriter.AddAttribute Method

Pascal

```
procedure AddAttribute(Area: TPcktAttrArea; const ParamName: string; const Value: OleVariant;
IncludeInDelta: Boolean); virtual;
```

C++

```
virtual __fastcall AddAttribute(TPcktAttrArea Area, const UnicodeString ParamName, const
OleVariant Value, Boolean IncludeInDelta);
```

Description

This is AddAttribute, a member of class TCustomPacketWriter.

58.1.3.5.2 TCustomPacketWriter.Check Method

Pascal

```
procedure Check(Status: Integer);
```

C++

```
__fastcall Check(int Status);
```

Description

This is Check, a member of class TCustomPacketWriter.

58.1.3.5.3 TCustomPacketWriter.SetDSWriter Method

Pascal

```
procedure SetDSWriter(const Value: IDSWriter);
```

C++

```
__fastcall SetDSWriter(const IDSWriter Value);
```

Description

This is SetDSWriter, a member of class TCustomPacketWriter.

58.1.3.6 TCustomPacketWriter Properties

The properties of the TCustomPacketWriter class are listed here.

Protected Properties

	Name	Description
	DSWriter (see page 3096)	This is DSWriter, a member of class TCustomPacketWriter.

58.1.3.6.1 TCustomPacketWriter.DSWriter Property

Pascal

```
property DSWriter: IDSWriter;
```

C++

```
__property IDSWriter DSWriter;
```

Description

This is DSWriter, a member of class TCustomPacketWriter.

58.1.4 TCustomProvider Class

TCustomProvider is the base class for data providers.

Class Hierarchy

Pascal

```
TCustomProvider = class(TComponent);
```

C++

```
class TCustomProvider : public TComponent;
```

File

Provider

Description

TCustomProvider defines the fundamental properties, methods, and events required of all provider objects, either through abstract (in C++ terminology, pure virutal) method declarations, or through virtual methods that do nothing but provide a backstop.

Do not create instances of TCustomProvider. To add a provider component to a form or data module, use a descendant of TCustomProvider such as TDataSetProvider (see page 3145) or TXMLTransformProvider.

A provider derived from TCustomProvider is an intermediate link between TClientDataSet or TXMLBroker object and a source dataset or source XML document. The source dataset or XML document can reside in the same application as the client dataset or XML broker, or it can be part of a separate application server. The provider packages data from the source dataset or XML document and passes it in one or more transportable data packets to the client dataset or XML broker. The client dataset reconstructs the data in the data packet to create a local, in-memory copy for user access. When user access is complete, the client dataset repackages any changed data and sends the updates back to the provider. The provider applies the updates to the database or source dataset.

When the provider supplies data to a client dataset, the client dataset reconstructs the data in the data packet to create a local, in-memory copy for user access. When user access is complete, the client dataset repackages any changed data and sends the updates back to the provider. The provider applies the updates to the database, to the source dataset, or to the source XML document.

If the provider can encode data packets in XML, it can supply data to an XML broker, which adds the data packet to an HTML document that is downloaded by a Web client. When the XML broker receives updates from the Web client, it sends the updates to the provider, which applies them to the database, to the source dataset, or to the source XML document.

Client datasets and XML brokers communicate with a provider through the IAppServer interface. In multi-tiered applications, this is the interface of the remote data module that contains the provider. To enable the remote data module to pass calls to the provider, the provider's Exported (see page 3115) property must be true, and its Owner property must specify the remote data module.

See Also

TClientDataSet

TDataSetProvider (see page 3145)

IAppServer

TXMLTransformProvider

58.1.4.1 TCustomProvider Members

The following tables list the members exposed by TCustomProvider.

Protected Events

	Name	Description
	AfterApplyUpdates (see page 3116)	Occurs after the provider applies updates.
	AfterExecute (see page 3117)	Occurs after the provider passes an execute command to its dataset.
	AfterGetParams (see page 3118)	Occurs after the provider creates an OleVariant containing parameter values to send to the client dataset.
	AfterGetRecords (see page 3118)	Occurs after the provider creates a data packet to send to the client dataset.
	AfterRowRequest (see page 3119)	Occurs after the provider fetches new information about the current record.
	BeforeApplyUpdates (see page 3120)	Occurs before the provider applies updates from a client dataset.
	BeforeExecute (see page 3121)	Occurs before the provider passes an execute command to its dataset.
	BeforeGetParams (see page 3121)	Occurs before the provider creates an OleVariant that encodes the current parameter values and sends them to a client dataset.

	BeforeGetRecords (see page 3122)	Occurs before the provider creates a data packet and sends it to a client dataset.
	BeforeRowRequest (see page 3123)	Occurs before the provider creates a delta packet with information about the current record.
	OnDataRequest (see page 3124)	Occurs when a provider calls its DataRequest (see page 3101) method.
	OnValidate (see page 3125)	This is OnValidate, a member of class TCustomProvider.

Protected Methods

	Name	Description
V	DoAfterApplyUpdates (see page 3102)	Generates an AfterApplyUpdates (see page 3116) event.
V	DoAfterExecute (see page 3102)	Generates an AfterExecute (see page 3117) event.
V	DoAfterGetParams (see page 3102)	Generates an AfterGetParams (see page 3118) event.
V	DoAfterGetRecords (see page 3103)	Generates an AfterGetRecords (see page 3118) event.
V	DoAfterRowRequest (see page 3103)	Generates an AfterRowRequest (see page 3119) event.
V	DoApplyUpdates (see page 3104)	This is DoApplyUpdates, a member of class TCustomProvider.
V	DoBeforeApplyUpdates (see page 3104)	Generates a BeforeApplyUpdates (see page 3120) event.
V	DoBeforeExecute (see page 3105)	Generates a BeforeExecute (see page 3121) event.
V	DoBeforeGetParams (see page 3105)	Generates a BeforeGetParams (see page 3121) event.
V	DoBeforeGetRecords (see page 3106)	Generates a BeforeGetRecords (see page 3122) event.
V	DoBeforeRowRequest (see page 3106)	Generates a BeforeRowRequest (see page 3123) event.
V	DoGetRecords (see page 3107)	This is DoGetRecords, a member of class TCustomProvider.
A	InternalApplyUpdates (see page 3110)	Provides the underlying implementation of the ApplyUpdates (see page 3100) method.
V	InternalExecute (see page 3111)	Provides the underlying implementation of the Execute (see page 3107) method.
V	InternalGetParams (see page 3111)	Returns the requested parameter values as a Variant array.
V	InternalGetRecords (see page 3112)	Provides the underlying implementation of the GetRecords (see page 3109) method.
V	InternalRowRequest (see page 3113)	Provides the underlying implementation of the RowRequest (see page 3113) method.
	SetActiveUpdateException (see page 3114)	This is SetActiveUpdateException, a member of class TCustomProvider.

Protected Properties

	Name	Description
R	ActiveUpdateException (see page 3114)	This is ActiveUpdateException, a member of class TCustomProvider.
	OwnerData (see page 3115)	This is OwnerData, a member of class TCustomProvider.

Public Methods

	Name	Description
	ApplyUpdates (see page 3100)	This is the overview for the ApplyUpdates method overload.
V	Create (see page 3099)	Creates a new TCustomProvider (see page 3096) object.
V	DataRequest (see page 3101)	Generates an OnDataRequest (see page 3124) event.
V	Destroy (see page 3099)	Frees an instance of a provider object.
V	Execute (see page 3107)	Executes a query, stored procedure, or custom SQL statement.
V	GetParams (see page 3108)	Returns the current parameter values.
	GetRecords (see page 3109)	This is the overview for the GetRecords method overload.
V	RowRequest (see page 3113)	Returns information from a specified record.

Public Properties

	Name	Description
R	Data (see page 3115)	Denotes a data packet created by the provider.
	Exported (see page 3115)	Allows client applications to pass calls to the provider when it resides on an application server.

58.1.4.2 TCustomProvider.Create Constructor

Creates a new TCustomProvider (see page 3096) object.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomProvider(TComponent * AOwner);
```

Description

Call Create to instantiate a provider at runtime. Providers placed in forms or data modules at design time are created automatically.

The constructor assigns the AOwner parameter as the owner of the provider. This owner is responsible for freeing the provider when it is destroyed. If the owner is a remote data module, the constructor registers the provider with that remote data module so that it is available to client applications through the IAppServer interface.

See Also

IAppServer

Providers

RegisterProvider

Owner

58.1.4.3 TCustomProvider.Destroy Destructor

Frees an instance of a provider object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomProvider();
```

Description

Do not call Destroy directly. Instead, call Free, which checks that the provider is not nil before calling its Destroy method. If the provider's Owner is a remote data module, Destroy unregisters the provider from that remote data module.

58.1.4.4 TCustomProvider Methods

The methods of the TCustomProvider class are listed here.

Protected Methods

	Name	Description
	DoAfterApplyUpdates (see page 3102)	Generates an AfterApplyUpdates (see page 3116) event.
	DoAfterExecute (see page 3102)	Generates an AfterExecute (see page 3117) event.
	DoAfterGetParams (see page 3102)	Generates an AfterGetParams (see page 3118) event.
	DoAfterGetRecords (see page 3103)	Generates an AfterGetRecords (see page 3118) event.
	DoAfterRowRequest (see page 3103)	Generates an AfterRowRequest (see page 3119) event.

	DoApplyUpdates (see page 3104)	This is DoApplyUpdates, a member of class TCustomProvider.
	DoBeforeApplyUpdates (see page 3104)	Generates a BeforeApplyUpdates (see page 3120) event.
	DoBeforeExecute (see page 3105)	Generates a BeforeExecute (see page 3121) event.
	DoBeforeGetParams (see page 3105)	Generates a BeforeGetParams (see page 3121) event.
	DoBeforeGetRecords (see page 3106)	Generates a BeforeGetRecords (see page 3122) event.
	DoBeforeRowRequest (see page 3106)	Generates a BeforeRowRequest (see page 3123) event.
	DoGetRecords (see page 3107)	This is DoGetRecords, a member of class TCustomProvider.
	InternalApplyUpdates (see page 3110)	Provides the underlying implementation of the ApplyUpdates (see page 3100) method.
	InternalExecute (see page 3111)	Provides the underlying implementation of the Execute (see page 3107) method.
	InternalGetParams (see page 3111)	Returns the requested parameter values as a Variant array.
	InternalGetRecords (see page 3112)	Provides the underlying implementation of the GetRecords (see page 3109) method.
	InternalRowRequest (see page 3113)	Provides the underlying implementation of the RowRequest (see page 3113) method.
	SetActiveUpdateException (see page 3114)	This is SetActiveUpdateException, a member of class TCustomProvider.

Public Methods

	Name	Description
	ApplyUpdates (see page 3100)	This is the overview for the ApplyUpdates method overload.
	DataRequest (see page 3101)	Generates an OnDataRequest (see page 3124) event.
	Execute (see page 3107)	Executes a query, stored procedure, or custom SQL statement.
	GetParams (see page 3108)	Returns the current parameter values.
	GetRecords (see page 3109)	This is the overview for the GetRecords method overload.
	RowRequest (see page 3113)	Returns information from a specified record.

58.1.4.4.1 ApplyUpdates Method

This is the overview for the ApplyUpdates method overload.

Overload List

	Name	Description
	TCustomProvider.ApplyUpdates (OleVariant, Integer, Integer) (see page 3100)	Applies updates received as a delta packet.
	TCustomProvider.ApplyUpdates (OleVariant, Integer, Integer, OleVariant) (see page 3100)	Applies updates received as a delta packet.

58.1.4.4.1.1 TCustomProvider.ApplyUpdates Method (OleVariant, Integer, Integer)

Applies updates received as a delta packet.

Pascal

```
function ApplyUpdates(const Delta: OleVariant; MaxErrors: Integer; out ErrorCount: Integer):
OleVariant; overload;
function ApplyUpdates(const Delta: OleVariant; MaxErrors: Integer; out ErrorCount: Integer;
var OwnerData: OleVariant): OleVariant; overload;
```

C++

```
__fastcall OleVariant ApplyUpdates(const OleVariant Delta, int MaxErrors, int ErrorCount);
__fastcall OleVariant ApplyUpdates(const OleVariant Delta, int MaxErrors, int ErrorCount,
OleVariant OwnerData);
```

Description

Call ApplyUpdates to apply updates contained in a delta packet. ApplyUpdates generates a BeforeApplyUpdates (see page

3120) event, applies the updates, and then generates an AfterApplyUpdates (see page 3116) event.

Specify the delta packet as the Delta parameter.

Use the MaxErrors parameter to indicate the maximum number of errors that can be applied before stopping the update operation. If MaxErrors is –1, the provider tries to apply all updates.

ErrorCount returns the number of errors encountered during the update operation.

The OwnerData (see page 3115) parameter (if used) supplies custom information that is passed to the BeforeApplyUpdates (see page 3120) event handler and which returns any information supplied by that event handler.

ApplyUpdates returns a data packet (as an OleVariant) of updates that could not be applied to the database.

Note: ApplyUpdates always generates the BeforeApplyUpdates (see page 3120) and AfterApplyUpdates (see page 3116) events, even if you use the syntax that does not include an OwnerData (see page 3115) parameter. In that case, the value of OwnerData (see page 3115) is nil (Delphi) or NULL (C++) on entry to the BeforeApplyUpdates (see page 3120) event handler and is discarded on exit from the AfterApplyUpdates (see page 3116) event handler.

Warning: The provider can't detect when an update conflicts with another user's changes to a memo field. This situation will not contribute to the errors counted by ErrorCount.

See Also

BeforeApplyUpdates (see page 3120)

AfterApplyUpdates (see page 3116)

AS_ApplyUpdates

ApplyUpdates

InternalApplyUpdates (see page 3110)

58.1.4.4.2 TCustomProvider.DataRequest Method

Generates an OnDataRequest (see page 3124) event.

Pascal

```
function DataRequest(Input: OleVariant): OleVariant; virtual;
```

C++

```
virtual __fastcall OleVariant DataRequest(OleVariant Input);
```

Description

Write an OnDataRequest (see page 3124) event handler and call the DataRequest function to invoke the event handler at runtime. This function and the event it invokes enable an application to perform custom processing in response to a request for data.

Note: Calling the client dataset's DataRequest method causes a call to the provider's DataRequest method.

See Also

OnDataRequest (see page 3124)

AS_DataRequest

DataRequest

Communicating with the Client Dataset

58.1.4.4.3 TCustomProvider.DoAfterApplyUpdates Method

Generates an AfterApplyUpdates (see page 3116) event.

Pascal

```
procedure DoAfterApplyUpdates(var OwnerData: OleVariant); virtual;
```

C++

```
virtual __fastcall DoAfterApplyUpdates(OleVariant OwnerData);
```

Description

The ApplyUpdates (see page 3100) method calls DoAfterApplyUpdates to generate the AfterApplyUpdates (see page 3116) event after applying updates and resolving any update errors that are handled by the provider.

OwnerData (see page 3115) supplies custom information to the AfterApplyUpdates (see page 3116) event handler. This is the same as the outgoing OwnerData (see page 3115) parameter of the BeforeApplyUpdates (see page 3120) event handler. The AfterApplyUpdates (see page 3116) event handler can change this value. The returned value is then passed to the client if it supplied an OwnerData (see page 3115) parameter when calling ApplyUpdates (see page 3100).

See Also

ApplyUpdates (see page 3100)

AfterApplyUpdates (see page 3116)

BeforeApplyUpdates (see page 3120)

58.1.4.4.4 TCustomProvider.DoAfterExecute Method

Generates an AfterExecute (see page 3117) event.

Pascal

```
procedure DoAfterExecute(var OwnerData: OleVariant); virtual;
```

C++

```
virtual __fastcall DoAfterExecute(OleVariant OwnerData);
```

Description

The Execute (see page 3107) method calls DoAfterExecute to generate the AfterExecute (see page 3117) event after executing CommandText or the associated query or stored procedure but before fetching any output parameters.

OwnerData (see page 3115) supplies custom information to the AfterExecute (see page 3117) event handler. This is the same as the outgoing OwnerData (see page 3115) parameter of the BeforeExecute (see page 3121) event handler. The AfterExecute (see page 3117) event handler can change this value. The returned value is then passed to the client.

See Also

Execute (see page 3107)

AfterExecute (see page 3117)

BeforeExecute (see page 3121)

58.1.4.4.5 TCustomProvider.DoAfterGetParams Method

Generates an AfterGetParams (see page 3118) event.

Pascal

```
procedure DoAfterGetParams(var OwnerData: OleVariant); virtual;
```

C++

```
virtual __fastcall DoAfterGetParams(OleVariant OwnerData);
```

Description

The GetParams (see page 3108) method calls DoAfterGetParams to generate the AfterGetParams (see page 3118) event after fetching the current parameter values of the associated dataset.

OwnerData (see page 3115) supplies custom information to the AfterGetParams (see page 3118) event handler. This is the same as the outgoing OwnerData (see page 3115) parameter of the BeforeGetParams (see page 3121) event handler. The AfterGetParams (see page 3118) event handler can change this value. The returned value is then passed to the client.

See Also

GetParams (see page 3108)

AfterGetParams (see page 3118)

BeforeGetParams (see page 3121)

58.1.4.4.6 TCustomProvider.DoAfterGetRecords Method

Generates an AfterGetRecords (see page 3118) event.

Pascal

```
procedure DoAfterGetRecords(var OwnerData: OleVariant); virtual;
```

C++

```
virtual __fastcall DoAfterGetRecords(OleVariant OwnerData);
```

Description

The GetRecords (see page 3109) method calls DoAfterGetRecords to generate the AfterGetRecords (see page 3118) event after assembling a data packet but before updating the parameters to reflect any output parameters.

OwnerData (see page 3115) supplies custom information to the AfterGetRecords (see page 3118) event handler. This is the same as the outgoing OwnerData (see page 3115) parameter of the BeforeGetRecords (see page 3122) event handler. The AfterGetRecords (see page 3118) event handler can change this value. The returned value is then passed to the client.

See Also

GetRecords (see page 3109)

AfterGetRecords (see page 3118)

BeforeGetRecords (see page 3122)

58.1.4.4.7 TCustomProvider.DoAfterRowRequest Method

Generates an AfterRowRequest (see page 3119) event.

Pascal

```
procedure DoAfterRowRequest(var OwnerData: OleVariant); virtual;
```

C++

```
virtual __fastcall DoAfterRowRequest(OleVariant OwnerData);
```

Description

The RowRequest (see page 3113) method calls DoAfterRowRequest to generate the AfterRowRequest (see page 3119) event after fetching the requested data.

OwnerData (see page 3115) supplies custom information to the AfterRowRequest (see page 3119) event handler. This is the same as the outgoing OwnerData (see page 3115) parameter of the BeforeRowRequest (see page 3123) event handler. The AfterRowRequest (see page 3119) event handler can change this value. The returned value is then passed to the client.

See Also

RowRequest (see page 3113)

AfterRowRequest (see page 3119)

BeforeRowRequest (see page 3123)

58.1.4.4.8 TCustomProvider.DoApplyUpdates Method

Pascal

```
function DoApplyUpdates(const Delta: OleVariant; MaxErrors: Integer; out ErrorCount: Integer;
var OwnerData: OleVariant): OleVariant; virtual;
```

C++

```
virtual __fastcall OleVariant DoApplyUpdates(const OleVariant Delta, int MaxErrors, int
ErrorCount, OleVariant OwnerData);
```

Description

This is DoApplyUpdates, a member of class TCustomProvider.

58.1.4.4.9 TCustomProvider.DoBeforeApplyUpdates Method

Generates a BeforeApplyUpdates (see page 3120) event.

Pascal

```
procedure DoBeforeApplyUpdates(var OwnerData: OleVariant); virtual;
```

C++

```
virtual __fastcall DoBeforeApplyUpdates(OleVariant OwnerData);
```

Description

The ApplyUpdates (see page 3100) method calls DoBeforeApplyUpdates to generate the BeforeApplyUpdates (see page 3120) event before it applies the updates in Delta.

Delta is the delta packet containing the updates as received from the client.

MaxErrors is the number of errors tolerated before the update operation should aborted.

OwnerData (see page 3115) supplies custom information to the BeforeApplyUpdates (see page 3120) event handler. The BeforeApplyUpdates (see page 3120) event handler can change this value. The returned value is then passed to the AfterApplyUpdates (see page 3116) event handler after applying the updates in Delta.

As implemented in TCustomProvider (see page 3096), Delta and MaxErrors are ignored.

See Also

ApplyUpdates (see page 3100)

AfterApplyUpdates (see page 3116)

BeforeApplyUpdates (see page 3120)

58.1.4.4.10 TCustomProvider.DoBeforeExecute Method

Generates a BeforeExecute (see page 3121) event.

Pascal

```
procedure DoBeforeExecute(const CommandText: WideString; var Params: OleVariant; var
OwnerData: OleVariant); virtual;
```

C++

```
virtual __fastcall DoBeforeExecute(const BSTR CommandText, OleVariant Params, OleVariant
OwnerData);
```

Description

The Execute (see page 3107) method calls DoBeforeExecute to generate the BeforeExecute (see page 3121) event before it executes CommandText or executes an associated query or stored procedure.

CommandText specifies an optional SQL statement that replaces the SQL of an associated query, or the name of a stored procedure that replaces the associated stored procedure. This parameter is ignored if the Options property does not include poAllowCommandText.

Params contains any parameter values that should be applied to CommandText or to the query or stored procedure before execution.

OwnerData (see page 3115) supplies custom information to the BeforeExecute (see page 3121) event handler. The event handler can change this value. The returned value is then passed to an AfterExecute (see page 3117) event handler and then returned to the caller of the Execute (see page 3107) method.

As implemented in TCustomProvider (see page 3096), CommandText and Params are ignored. Descendant classes apply these values before generating the BeforeExecute (see page 3121) event.

See Also

Execute (see page 3107)

BeforeExecute (see page 3121)

AfterExecute (see page 3117)

Options (see page 3087)

58.1.4.4.11 TCustomProvider.DoBeforeGetParams Method

Generates a BeforeGetParams (see page 3121) event.

Pascal

```
procedure DoBeforeGetParams(var OwnerData: OleVariant); virtual;
```

C++

```
virtual __fastcall DoBeforeGetParams(OleVariant OwnerData);
```

Description

The GetParams (see page 3108) method calls DoBeforeGetParams to generate the BeforeGetParams (see page 3121) event before fetching the current parameter values of the associated dataset.

OwnerData (see page 3115) supplies custom information to the BeforeGetParams (see page 3121) event handler. The event handler can change this value. The returned value is then passed to an AfterGetParams (see page 3118) event handler

and then returned to the caller of the GetParams (see page 3108) method.

See Also

GetParams (see page 3108)

BeforeGetParams (see page 3121)

AfterGetParams (see page 3118)

58.1.4.4.12 TCustomProvider.DoBeforeGetRecords Method

Generates a BeforeGetRecords (see page 3122) event.

Pascal

```
procedure DoBeforeGetRecords(Count: Integer; Options: Integer; const CommandText: WideString;
var Params: OleVariant; var OwnerData: OleVariant); virtual;
```

C++

```
virtual __fastcall DoBeforeGetRecords(int Count, int Options, const BSTR CommandText,
OleVariant Params, OleVariant OwnerData);
```

Description

The GetRecords (see page 3109) method calls DoBeforeGetRecords to generate the BeforeGetRecords (see page 3122) event before it fetches data and assembles it into a data packet.

Count is the number of records requested. If Count is –1, all records are requested.

Options indicates whether the data packet should include metadata, whether it should start with the first record, and whether the data packet should be encoded in XML. For details, see the GetRecords (see page 3109) method.

CommandText specifies an optional SQL statement that replaces the SQL of an associated query, or the name of a table or stored procedure that replaces the associated table or stored procedure. This parameter is ignored if the Options property does not include poAllowCommandText or if the provider does not have an Options property.

Params contains any parameter values that should be applied to CommandText or to a query or stored procedure before fetching data into a data packet.

OwnerData (see page 3115) supplies custom information to the BeforeGetRecords (see page 3122) event handler. The event handler can change this value. The returned value is then passed to an AfterGetRecords (see page 3118) event handler and then (optionally) returned to the caller of the GetRecords (see page 3109) method.

As implemented in TCustomProvider (see page 3096), Count, Options, CommandText, and Params are ignored. Descendant classes can apply these values before generating the BeforeExecute (see page 3121) event.

See Also

GetRecords (see page 3109)

BeforeGetRecords (see page 3122)

AfterGetRecords (see page 3118)

Options (see page 3087)

58.1.4.4.13 TCustomProvider.DoBeforeRowRequest Method

Generates a BeforeRowRequest (see page 3123) event.

Pascal

```
procedure DoBeforeRowRequest(var OwnerData: OleVariant); virtual;
```

C++

```
virtual __fastcall DoBeforeRowRequest(OleVariant OwnerData);
```

Description

The RowRequest (see page 3113) method calls DoBeforeRowRequest to generate the BeforeRowRequest (see page 3123) event before fetching the requested data.

OwnerData (see page 3115) supplies custom information to the BeforeRowRequest (see page 3123) event handler. The event handler can change this value. The returned value is then passed to an AfterRowRequest (see page 3119) event handler and then returned to the caller of the RowRequest (see page 3113) method.

See Also

RowRequest (see page 3113)

BeforeRowRequest (see page 3123)

AfterRowRequest (see page 3119)

58.1.4.4.14 TCustomProvider.DoGetRecords Method

Pascal

```
function DoGetRecords(Count: Integer; out RecsOut: Integer; Options: Integer; const
CommandText: WideString; var Params: OleVariant; var OwnerData: OleVariant): OleVariant;
virtual;
```

C++

```
virtual __fastcall OleVariant DoGetRecords(int Count, int RecsOut, int Options, const BSTR
CommandText, OleVariant Params, OleVariant OwnerData);
```

Description

This is DoGetRecords, a member of class TCustomProvider.

58.1.4.4.15 TCustomProvider.Execute Method

Executes a query, stored procedure, or custom SQL statement.

Pascal

```
procedure Execute(const CommandText: WideString; var Params: OleVariant; var OwnerData:
OleVariant); virtual;
```

C++

```
virtual __fastcall Execute(const BSTR CommandText, OleVariant Params, OleVariant OwnerData);
```

Description

Client datasets call Execute through the AS_Execute method of the IAppServer interface. In response, the provider executes an SQL command. If the provider represents a query or stored procedure that does not return a cursor, calling Execute causes the provider to execute the query or stored procedure. If the caller supplies a value for CommandText, Execute may execute that statement instead (see below).

Before executing the appropriate command, Execute generates a BeforeExecute (see page 3121) event. After executing the command, it generates an AfterExecute (see page 3117) event. Finally, it returns any output parameters as the value of the Params parameter.

CommandText is an SQL statement that replaces the SQL of an associated query, or the name of a stored procedure that replaces the associated stored procedure. This parameter is ignored if the Options property does not include poAllowCommandText.

On entry, Params supplies parameter values for the query, stored procedure, or CommandText string. On exit, Params returns any output parameters.

OwnerData (see page 3115) contains custom information passed to the provider from the client dataset's BeforeExecute (see page 3121) event handler. This information is passed in to the provider's BeforeExecute (see page 3121) event handler and is returned by the provider's AfterExecute (see page 3117) event handler.

See Also

AS_Execute

BeforeExecute (see page 3121)

AfterExecute (see page 3117)

BeforeExecute

AfterExecute

Execute

CommandText

InternalExecute (see page 3111)

Options (see page 3087)

Communicating with the Client Dataset

58.1.4.4.16 TCustomProvider.GetParams Method

Returns the current parameter values.

Pascal

```
function GetParams(var OwnerData: OleVariant): OleVariant; virtual;
```

C++

```
virtual __fastcall OleVariant GetParams(OleVariant OwnerData);
```

Description

GetParams returns a variant array of variant arrays that contains the current parameter values for the provider's data source. The first array dimension is an index into all the parameter values. Its value runs from 0 to one less than the number of parameters. The second array dimension has four values: The parameter name (a Delphi string or a C++ AnsiString), the parameter data type (a TFieldType), the parameter type (a TParamType), and the parameter value (a Variant). The values in this array can be applied to a TParams object using the global UnpackParams procedure.

Before fetching the current parameter values, GetParams generates a BeforeGetParams (see page 3121) event, passing the value supplied as the OwnerData (see page 3115) parameter to the event handler. After creating the OleVariant that contains the parameter values, GetParams generates an AfterGetParams (see page 3118) event, passing in the current value of OwnerData (see page 3115). Finally, the value of OwnerData (see page 3115) returned by the AfterGetParams (see page 3118) is returned in the OwnerData (see page 3115) parameter.

See Also

FetchParams

ParamType

Name

DataType

Value

AS_GetParams

UnpackParams

BeforeGetParams (see page 3121)

AfterGetParams (see page 3118)

InternalGetParams (see page 3111)

Communicating with the Client Dataset

58.1.4.4.17 GetRecords Method

This is the overview for the GetRecords method overload.

Overload List

	Name	Description
	TCustomProvider.GetRecords (Integer, Integer, Integer) (see page 3109)	Retrieves a data packet that represents the provider's associated data.
	TCustomProvider.GetRecords (Integer, Integer, Integer, WideString, OleVariant, OleVariant) (see page 3109)	Retrieves a data packet that represents the provider's associated data.

58.1.4.4.17.1 TCustomProvider.GetRecords Method (Integer, Integer, Integer)

Retrieves a data packet that represents the provider's associated data.

Pascal

```
function GetRecords(Count: Integer; out RecsOut: Integer; Options: Integer): OleVariant;
overload;
function GetRecords(Count: Integer; out RecsOut: Integer; Options: Integer; const CommandText:
WideString; var Params: OleVariant; var OwnerData: OleVariant): OleVariant; overload;
```

C++

```
__fastcall OleVariant GetRecords(int Count, int RecsOut, int Options);
__fastcall OleVariant GetRecords(int Count, int RecsOut, int Options, const BSTR CommandText,
OleVariant Params, OleVariant OwnerData);
```

Description

Call GetRecords to retrieve a data packet containing a specified number of records. GetRecords generates a BeforeGetRecords (see page 3122) event, assembles a data packet, and then generates an AfterGetRecords (see page 3118) event.

Count indicates the number of records to retrieve. If Count is -1, all records are retrieved. If Count is 0, only metadata is retrieved. If Count is greater than 0, only the specified number of records are retrieved.

RecsOut returns the actual number of records in the data packet.

Options indicates what information should be added to the data packet in addition to data. It is a combination of the GetRecordOption constants. For example, the following value requests metadata along with a data packet encoded in XML:

```
MetaDataOption + XMLOption
MetaDataOption || XMLOption
```

CommandText specifies an optional SQL statement that replaces the SQL of an associated query, or the name of a table or stored procedure that replaces the associated table or stored procedure. This parameter is ignored if the Options property does not include poAllowCommandText or if the provider does not have an Options property.

Params specifies any parameter values that should be applied to CommandText or to an associated query or stored procedure. If the provider is associated with a TTable, Params specifies field values that filter the result set. On exit from GetRecords, Params returns any output parameters.

OwnerData (see page 3115) contains custom information that is supplied by a client dataset's BeforeGetRecords (see page 3122) event handler. This information is passed to the provider's BeforeGetRecords (see page 3122) event handler and is returned by the provider's AfterGetRecords (see page 3118) event handler.

Records are returned as a data packet in a Variant.

Note: GetRecords always generates the BeforeGetRecords (see page 3122) and AfterGetRecords (see page 3118) events, even if you use the syntax that does not include an OwnerData (see page 3115) parameter. In that case, the value of OwnerData (see page 3115) is nil (Delphi) or NULL (C++) on entry to the BeforeGetRecords (see page 3122) event handler and is discarded on exit from the AfterGetRecords (see page 3118) event handler.

Tip: If the provider is part of a stateless application server, there is no guarantee that the provider will be positioned on the desired record before it creates a data packet. Use a BeforeGetRecords (see page 3122) event handler to position the cursor before the call to GetRecords.

See Also

DataRequest (see page 3101)

Data (see page 3115)

BeforeGetRecords (see page 3122)

AS_GetRecords

AfterGetRecords (see page 3118)

InternalGetRecords (see page 3112)

BeforeGetRecords

Communicating with the Client Dataset

Responding to Client Data Requests

58.1.4.4.18 TCustomProvider.InternalApplyUpdates Method

Provides the underlying implementation of the ApplyUpdates (see page 3100) method.

Pascal

```
function InternalApplyUpdates(const Delta: OleVariant; MaxErrors: Integer; out ErrorCount:
Integer): OleVariant; virtual; abstract;
```

C++

```
virtual __fastcall OleVariant InternalApplyUpdates(const OleVariant Delta, int MaxErrors, int
ErrorCount) = 0;
```

Description

The ApplyUpdates (see page 3100) method calls InternalApplyUpdates to apply the updates contained in a delta packet after generating the BeforeApplyUpdates (see page 3120) event and before generating the AfterApplyUpdates (see page 3116) event. The InternalApplyUpdates method of TCustomProvider (see page 3096) is abstract (Delphi) or pure virtual (C++),

meaning that it has no implementation. Descendant classes override this method to apply updates, resolve update errors, and return any updates that could not be applied.

Delta is the delta packet containing insertions, deletions, and modifications to be applied.

MaxErrors indicates the maximum number of errors that can be applied before stopping the update operation. If MaxErrors is –1, the provider tries to apply all updates.

ErrorCount returns the number of errors encountered during the update operation.

InternalApplyUpdates returns a data packet of updates that could not be applied to the database as an OleVariant.

See Also

ApplyUpdates (see page 3100)

58.1.4.4.19 TCustomProvider.InternalExecute Method

Provides the underlying implementation of the Execute (see page 3107) method.

Pascal

```
procedure InternalExecute(const CommandText: WideString; var Params: OleVariant); virtual;
```

C++

```
virtual __fastcall InternalExecute(const BSTR CommandText, OleVariant Params);
```

Description

The Execute (see page 3107) method calls InternalExecute to execute the appropriate SQL statement after calling DoBeforeExecute (see page 3105) to generate a BeforeExecute (see page 3121) event and apply the CommandText and Params. As Implemented in TCustomProvider (see page 3096), InternalExecute does nothing. Descendants override this method to pass the execute command on to an associated dataset.

CommandText is an SQL statement that replaces the SQL of an associated query, or the name of a stored procedure that replaces the associated stored procedure. This parameter is ignored if the Options property does not include poAllowCommandText.

On entry, Params supplies parameter values for the query, stored procedure, or CommandText string. On exit, Params returns any output parameters.

See Also

Execute (see page 3107)

DoBeforeExecute (see page 3105)

58.1.4.4.20 TCustomProvider.InternalGetParams Method

Returns the requested parameter values as a Variant array.

Pascal

```
function InternalGetParams(Types: TParamTypes = AllParamTypes): OleVariant; virtual;
```

C++

```
virtual __fastcall OleVariant InternalGetParams(TParamTypes Types = AllParamTypes);
```

Description

TCustomProvider (see page 3096) uses InternalGetParams internally whenever it needs to assemble the current parameter values into an OleVariant. This provides the underlying implementation of the GetParams (see page 3108) method, as well as

assembling parameter values for methods that return output parameters (such as Execute (see page 3107) or GetRecords (see page 3109)).

The Types parameter limits the parameter values that are returned to a subset of all possible parameter types. If this parameter is omitted, all parameter values are returned.

As implemented in TCustomProvider (see page 3096), InternalGetParams always returns NULL. Descendants override this method to return the requested parameter values.

See Also

GetParams (see page 3108)

Execute (see page 3107)

GetRecords (see page 3109)

58.1.4.4.21 TCustomProvider.InternalGetRecords Method

Provides the underlying implementation of the GetRecords (see page 3109) method.

Pascal

```
function InternalGetRecords(Count: Integer; out RecsOut: Integer; Options: TGetRecordOptions;
const CommandText: WideString; var Params: OleVariant): OleVariant; virtual;
```

C++

```
virtual __fastcall OleVariant InternalGetRecords(int Count, int RecsOut, TGetRecordOptions
Options, const BSTR CommandText, OleVariant Params);
```

Description

The GetRecords (see page 3109) method calls InternalGetRecords to assemble a data packet after calling DoBeforeGetRecords (see page 3106) to apply the CommandText and Params and generate a BeforeGetRecords (see page 3122) event. As implemented in TCustomProvider (see page 3096), InternalGetRecords always returns NULL. Descendant classes override this method to fetch the requested records and assemble them into a data packet.

Count indicates the number of records to retrieve. If Count is -1, all records are retrieved. If Count is 0, only metadata is retrieved. If Count is greater than 0, only the specified number of records are retrieved.

RecsOut returns the actual number of records retrieved.

Options indicates what information should be added to the data packet in addition to data. It can include grMetaData (include metadata), grReset (start providing records with the first one in the dataset), and grXML (encode the datapacket in XML rather than as an OleVariant) or grXMLUTF8 (same as grXML except extended characters are encoded using UTF8).

CommandText specifies an optional SQL statement that replaces the SQL of an associated query, or the name of a table or stored procedure that replaces the associated table or stored procedure. This parameter is ignored if the Options property does not include poAllowCommandText or if the provider does not have an Options property.

Params specifies any parameter values that should be applied to CommandText or to an associated query or stored procedure. If the provider is associated with a TTable object, Params specifies field values that filter the result set. On exit from GetRecords (see page 3109), Params returns any output parameters.

Records are returned as a data packet in a Variant.

See Also

GetRecords (see page 3109)

DoBeforeGetRecords (see page 3106)

58.1.4.4.22 TCustomProvider.InternalRowRequest Method

Provides the underlying implementation of the RowRequest (see page 3113) method.

Pascal

```
function InternalRowRequest(const Row: OleVariant; RequestType: TFetchOptions): OleVariant;
virtual;
```

C++

```
virtual __fastcall OleVariant InternalRowRequest(const OleVariant Row, TFetchOptions
RequestType);
```

Description

The RowRequest (see page 3113) method calls InternalRowRequest to assemble the specified information after generating a BeforeRowRequest (see page 3123) event and before generating an AfterRowRequest (see page 3119) event. As implemented in TCustomProvider (see page 3096), InternalRowRequest does nothing. Descendant classes override this method to assemble the requested information into an OleVariant.

The Row parameter is an OleVariant that describes the current record on the client dataset. It enables the provider to locate the record of interest.

InternalRowRequest returns the requested information in a delta packet.

See Also

RowRequest (see page 3113)

58.1.4.4.23 TCustomProvider.RowRequest Method

Returns information from a specified record.

Pascal

```
function RowRequest(const Row: OleVariant; RequestType: Integer; var OwnerData: OleVariant):
OleVariant; virtual;
```

C++

```
virtual __fastcall OleVariant RowRequest(const OleVariant Row, int RequestType, OleVariant
OwnerData);
```

Description

RowRequest is called automatically in response to the client dataset's FetchBlobs, FetchDetails, or RefreshRecord method. It generates a BeforeRowRequest (see page 3123) event, assembles the requested information, and generates an AfterRowRequest (see page 3119) event.

The Row parameter is an OleVariant that describes the record about which information is requested. It enables the provider to locate the record of interest.

The RequestType parameter indicates the type of information required. It is an integer version of the TFetchOptions type. (Its an integer because the value must be marshaled over an IAppServer interface). In Delphi, to convert the TFetchOptions value to the appropriate integer, use the Ord function. In C++, to combine TFetchOption values, use the bitwise "or" operator. For example:

```
Ord(foBlobs) + Ord(foDetails);
foBlobs || foDetails
```

RowRequest returns the requested information in a delta packet.

As implemented in TCustomProvider (see page 3096), RowRequest always returns NULL. Descendant classes override this method to return the requested data.

See Also

FetchBlobs

FetchDetails

RefreshRecord

BeforeRowRequest (see page 3123)

AfterRowRequest (see page 3119)

AS_RowRequest

InternalRowRequest (see page 3113)

Communicating with the Client Dataset

58.1.4.4.24 TCustomProvider.SetActiveUpdateException Method

Pascal

```
procedure SetActiveUpdateException(E: Exception);
```

C++

```
__fastcall SetActiveUpdateException(Exception E);
```

Description

This is SetActiveUpdateException, a member of class TCustomProvider.

58.1.4.5 TCustomProvider Properties

The properties of the TCustomProvider class are listed here.

Protected Properties

	Name	Description
	ActiveUpdateException (see page 3114)	This is ActiveUpdateException, a member of class TCustomProvider.
	OwnerData (see page 3115)	This is OwnerData, a member of class TCustomProvider.

Public Properties

	Name	Description
	Data (see page 3115)	Denotes a data packet created by the provider.
	Exported (see page 3115)	Allows client applications to pass calls to the provider when it resides on an application server.

58.1.4.5.1 TCustomProvider.ActiveUpdateException Property

Pascal

```
property ActiveUpdateException: Exception;
```

C++

```
__property Exception ActiveUpdateException;
```

Description

This is ActiveUpdateException, a member of class TCustomProvider.

58.1.4.5.2 TCustomProvider.Data Property

Denotes a data packet created by the provider.

Pascal

```
property Data: OleVariant;
```

C++

```
__property OleVariant Data;
```

Description

Data is a read-only property that provides access to a data packet containing all records. Reading Data calls GetRecords (see page 3109) with a parameter of -1.

As implemented in TCustomProvider (see page 3096), GetRecords (see page 3109) returns NULL. Descendants of TCustomProvider (see page 3096) reimplement the GetRecords (see page 3109) method to implement the Data property.

See Also

GetRecords (see page 3109)

58.1.4.5.3 TCustomProvider.Exported Property

Allows client applications to pass calls to the provider when it resides on an application server.

Pascal

```
property Exported: Boolean;
```

C++

```
__property Boolean Exported;
```

Description

Use Exported to indicate whether the provider is made available to clients of a remote data module that owns it. When Exported is true, client applications can specify the provider as a target for calls made using the IAppServer interface. When Exported is false, IAppServer calls that specify the provider cause an exception.

Note: In addition to the Exported property, the provider must be registered with the remote data module for clients to access it using the data module's IAppServer interface. When the provider's Owner is the remote data module, it is automatically registered. If the provider has a different owner, the server application must explicitly register providers using the remote data module's RegisterProvider method.

See Also

IAppServer

Providers

RegisterProvider

58.1.4.5.4 TCustomProvider.OwnerData Property

Pascal

```
property OwnerData: OleVariant;
```

C++

```
__property OleVariant OwnerData;
```

Description

This is OwnerData, a member of class TCustomProvider.

58.1.4.6 TCustomProvider Events

The events of the TCustomProvider class are listed here.

Protected Events

	Name	Description
	AfterApplyUpdates (see page 3116)	Occurs after the provider applies updates.
	AfterExecute (see page 3117)	Occurs after the provider passes an execute command to its dataset.
	AfterGetParams (see page 3118)	Occurs after the provider creates an OleVariant containing parameter values to send to the client dataset.
	AfterGetRecords (see page 3118)	Occurs after the provider creates a data packet to send to the client dataset.
	AfterRowRequest (see page 3119)	Occurs after the provider fetches new information about the current record.
	BeforeApplyUpdates (see page 3120)	Occurs before the provider applies updates from a client dataset.
	BeforeExecute (see page 3121)	Occurs before the provider passes an execute command to its dataset.
	BeforeGetParams (see page 3121)	Occurs before the provider creates an OleVariant that encodes the current parameter values and sends them to a client dataset.
	BeforeGetRecords (see page 3122)	Occurs before the provider creates a data packet and sends it to a client dataset.
	BeforeRowRequest (see page 3123)	Occurs before the provider creates a delta packet with information about the current record.
	OnDataRequest (see page 3124)	Occurs when a provider calls its DataRequest (see page 3101) method.
	OnValidate (see page 3125)	This is OnValidate, a member of class TCustomProvider.

58.1.4.6.1 TCustomProvider.AfterApplyUpdates Event

Occurs after the provider applies updates.

Pascal

```
property AfterApplyUpdates: TRemoteEvent;
```

C++

```
__property TRemoteEvent AfterApplyUpdates;
```

Description

Write an AfterApplyUpdates event handler to return custom information to a client dataset after applying updates. AfterApplyUpdates is part of the mechanism by which a client dataset and a provider communicate information about updates. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset calls the AS_ApplyUpdates method of the IAppServer interface, the following events occur:

1.The provider receives a BeforeApplyUpdates (see page 3120) event, where it can respond to or change custom information that the client dataset sends.

2.If it is a dataset provider, the provider receives an OnUpdateData event where it can edit the updates before they are applied.

3. The provider applies the updates, and attempts to resolve problems.

4.The provider receives an AfterApplyUpdates event, where it can encode custom information into its OwnerData (see page 3115) parameter or respond to information from the BeforeApplyUpdates (see page 3120) event handler.

5.The client dataset receives an AfterApplyUpdates event, where it can respond to the custom information returned by the

provider in its AfterApplyUpdates event handler.

Note: If ApplyUpdates (see page 3100) is called without the OwnerData (see page 3115) parameter, on entry to the event handler the OwnerData (see page 3115) parameter contains the value set in the BeforeApplyUpdates (see page 3120) event handler and it is ignored on exit from the AfterApplyUpdates event handler.

See Also

ApplyUpdates (see page 3100)

AS_ApplyUpdates

ApplyUpdates

AfterApplyUpdates

BeforeApplyUpdates (see page 3120)

OnUpdateData (see page 3092)

Responding to Client Update Requests

Communicating with the Client Dataset

58.1.4.6.2 TCustomProvider.AfterExecute Event

Occurs after the provider passes an execute command to its dataset.

Pascal

```
property AfterExecute: TRemoteEvent;
```

C++

```
__property TRemoteEvent AfterExecute;
```

Description

Write an AfterExecute event handler to return custom information to a client dataset after executing a query or stored procedure. AfterExecute is part of the mechanism by which a client dataset and a provider communicate information when executing queries or stored procedures that do not return a cursor. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset calls the AS_Execute method of the IAppServer interface, the following events occur:

1.The provider receives a BeforeExecute (see page 3121) event, where it can respond to or change custom information that the client dataset sends.

2.The provider dispatches its Execute (see page 3107) method to execute a query or stored procedure.

3.The provider receives an AfterExecute event, where it can encode custom information into its OwnerData (see page 3115) parameter or respond to information from the BeforeExecute (see page 3121) event handler.

4.The client dataset receives an AfterExecute event, where it can respond to the custom information returned by the provider in its AfterExecute event handler.

See Also

Execute (see page 3107)

AS_Execute

Execute

BeforeExecute (see page 3121)

DataSet (see page 3160)

AfterExecute

Communicating with the Client Dataset

58.1.4.6.3 TCustomProvider.AfterGetParams Event

Occurs after the provider creates an OleVariant containing parameter values to send to the client dataset.

Pascal

```
property AfterGetParams: TRemoteEvent;
```

C++

```
__property TRemoteEvent AfterGetParams;
```

Description

Write an AfterGetParams event handler to send custom information to a client dataset when returning parameters from the provider's dataset. AfterGetParams is part of the mechanism by which a provider and a client dataset communicate information about parameter values. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset calls the AS_GetParams method of the IAppServer interface, the following events occur:

1.The provider receives a BeforeGetParams (see page 3121) event, where it can respond to or change custom information that the client dataset sends.

2.The provider looks up the parameter values, if appropriate, and encodes them in an OleVariant.

3.The provider receives an AfterGetParams event, where it can encode custom information into its OwnerData (see page 3115) parameter or respond to information from the BeforeGetParams (see page 3121) event handler.

4.The client dataset receives an AfterGetParams event, where it can respond to the custom information returned by the provider in its AfterGetParams event handler.

See Also

AS_GetParams

GetParams (see page 3108)

AfterGetParams

BeforeGetParams (see page 3121)

58.1.4.6.4 TCustomProvider.AfterGetRecords Event

Occurs after the provider creates a data packet to send to the client dataset.

Pascal

```
property AfterGetRecords: TRemoteEvent;
```

C++

```
__property TRemoteEvent AfterGetRecords;
```

Description

Write an AfterGetRecords event handler to send custom information to a client dataset when providing a data packet. AfterGetRecords is part of the mechanism by which a provider and a client dataset communicate information about data fetching. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to

communicate persistent state information.

When the client dataset calls the AS_GetRecords method of the IAppServer interface, the following events occur:

1.The provider receives a BeforeGetRecords (see page 3122) event, where it can respond to or change custom information that the client dataset sends.

2.If it is a dataset provider, the provider receives an OnGetData event where it can edit the data before creating a data packet, and then generates the data packet.

3.The provider receives an AfterGetRecords event, where it can encode custom information into its OwnerData (see page 3115) parameter or respond to information from the BeforeGetRecords (see page 3122) event handler.

4.The client dataset receives an AfterGetRecords event, where it can respond to the custom information returned by the provider in its AfterGetRecords event handler.

See Also

AS_GetRecords

OnGetData (see page 3091)

GetRecords (see page 3109)

AfterGetRecords

BeforeGetRecords (see page 3122)

Communicating with the Client Dataset

Responding to Client Data Requests

58.1.4.6.5 TCustomProvider.AfterRowRequest Event

Occurs after the provider fetches new information about the current record.

Pascal

```
property AfterRowRequest: TRemoteEvent;
```

C++

```
__property TRemoteEvent AfterRowRequest;
```

Description

Write an AfterRowRequest event handler to send custom information to a client dataset along with information about the current record. AfterRowRequest is part of the mechanism by which a provider and a client dataset communicate information about data fetching. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset calls the AS_RowRequest method of the IAppServer interface, the following events occur:

1.The provider receives a BeforeRowRequest (see page 3123) event, where it can respond to or change custom information that the client dataset sends.

2.The provider creates a delta packet that contains the requested information.

3.The provider receives an AfterRowRequest event, where it can encode custom information into its OwnerData (see page 3115) parameter or respond to information from the BeforeRowRequest (see page 3123) event handler.

4.The client dataset receives an AfterRowRequest event, where it can respond to the custom information returned by the provider in its AfterRowRequest event handler.

See Also

AS_RowRequest

RowRequest (see page 3113)

AfterRowRequest

BeforeRowRequest (see page 3123)

FetchBlobs

FetchParams

RefreshRecord

Communicating with the Client Dataset

58.1.4.6.6 TCustomProvider.BeforeApplyUpdates Event

Occurs before the provider applies updates from a client dataset.

Pascal

```
property BeforeApplyUpdates: TRemoteEvent;
```

C++

```
__property TRemoteEvent BeforeApplyUpdates;
```

Description

Write a BeforeApplyUpdates event handler to respond to custom information from a client dataset. BeforeApplyUpdates is part of the mechanism by which a provider and a client dataset communicate information about updates. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset's ApplyUpdates (see page 3100) method is called, the following events occur:

1.The client dataset receives a BeforeApplyUpdates event, where it can encode custom information into an OleVariant.

2.The provider receives a BeforeApplyUpdates event, where the OleVariant from the client dataset appears as the OwnerData (see page 3115) parameter. The provider can respond to or change that information before applying the updates.

3.If it is a dataset provider, the provider applies the updates, and receives an OnUpdateError event for any problem records.

4.The provider receives an AfterApplyUpdates (see page 3116) event, where it can encode custom information into its OwnerData (see page 3115) parameter or respond to information from the BeforeApplyUpdates event handler.

5.The client dataset receives an AfterApplyUpdates (see page 3116) event, where it can respond to the custom information returned by the provider in its AfterApplyUpdates (see page 3116) event handler.

Note: If ApplyUpdates (see page 3100) is called without the OwnerData (see page 3115) parameter, the OwnerData (see page 3115) parameter is nil (Delphi) or NULL (C++) on entry to the event handler and its value (on exit) is only passed to the AfterApplyUpdates (see page 3116) event handler but not returned to the caller.

See Also

ApplyUpdates (see page 3100)

AS_ApplyUpdates

ApplyUpdates

BeforeApplyUpdates

AfterApplyUpdates (see page 3116)

Responding to Client Update Requests

Communicating with the Client Dataset

58.1.4.6.7 TCustomProvider.BeforeExecute Event

Occurs before the provider passes an execute command to its dataset.

Pascal

```
property BeforeExecute: TRemoteEvent;
```

C++

```
__property TRemoteEvent BeforeExecute;
```

Description

Write a BeforeExecute event handler to respond to custom information from a client dataset before executing a query or stored procedure. Any parameter values supplied by the client dataset with the Execute (see page 3107) command are applied before this event.

BeforeExecute is part of the mechanism by which a provider and a client dataset communicate information when executing queries or stored procedures that do not return cursors. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset's Execute (see page 3107) method is called, the following events occur:

1.The client dataset receives a BeforeExecute event, where it can encode custom information into an OleVariant.

2.The provider receives a BeforeExecute event, where the OleVariant from the client dataset appears as the OwnerData (see page 3115) parameter. The provider can respond to or change that information before passing the execute command on to its dataset.

3.The provider dispatches its Execute (see page 3107) method to execute a query or stored procedure.

4.The provider receives an AfterExecute (see page 3117) event, where it can encode custom information into its OwnerData (see page 3115) parameter or respond to information from the BeforeExecute event handler.

5.The client dataset receives an AfterExecute (see page 3117) event, where it can respond to the custom information returned by the provider in its AfterExecute (see page 3117) event handler.

See Also

Execute (see page 3107)

AS_Execute

Execute

DataSet (see page 3160)

BeforeExecute

58.1.4.6.8 TCustomProvider.BeforeGetParams Event

Occurs before the provider creates an OleVariant that encodes the current parameter values and sends them to a client dataset.

Pascal

```
property BeforeGetParams: TRemoteEvent;
```

C++

```
__property TRemoteEvent BeforeGetParams;
```

Description

Write a BeforeGetParams event handler to respond to custom information from a client dataset. BeforeGetParams is part of the mechanism by which a provider and a client dataset communicate information during parameter fetching. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset fetches parameter values from the provider, the following events occur:

1.The client dataset receives a BeforeGetParams event, where it can encode custom information into an OleVariant.

2.The provider receives a BeforeGetParams event, where the OleVariant from the client dataset appears as the OwnerData (see page 3115) parameter. The provider can respond to or change that information before it fetches and encodes the parameter values.

3.The provider generates an OleVariant for the current parameter values of its dataset, if appropriate.

4.The provider receives an AfterGetParams (see page 3118) event, where it can encode custom information into its OwnerData (see page 3115) parameter or respond to information from the BeforeGetParams event handler.

5.The client dataset receives an AfterGetParams (see page 3118) event, where it can respond to the custom information returned by the provider in its AfterGetParams (see page 3118) event handler.

See Also

AS_GetParams

GetParams (see page 3108)

BeforeGetParams

AfterGetParams (see page 3118)

Communicating with the Client Dataset

58.1.4.6.9 TCustomProvider.BeforeGetRecords Event

Occurs before the provider creates a data packet and sends it to a client dataset.

Pascal

```
property BeforeGetRecords: TRemoteEvent;
```

C++

```
__property TRemoteEvent BeforeGetRecords;
```

Description

Write a BeforeGetRecords event handler to respond to custom information from a client dataset. BeforeGetRecords is part of the mechanism by which a provider and a client dataset communicate information about data fetching. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset fetches data from the provider, the following events occur:

1.The client dataset receives a BeforeGetRecords event, where it can encode custom information into an OleVariant.

2.The provider receives a BeforeGetRecords event, where the OleVariant from the client dataset appears as the OwnerData (see page 3115) parameter. The provider can respond to or change that information before it creates a data packet.

3.The provider generates the data packet.

4.The provider receives an AfterGetRecords (see page 3118) event, where it can encode custom information into its OwnerData (see page 3115) parameter or respond to information from the BeforeGetRecords event handler.

5.The client dataset receives an AfterGetRecords (see page 3118) event, where it can respond to the custom information returned by the provider in its AfterGetRecords (see page 3118) event handler.

Delphi Examples:

```
{
This example shows the implementation of a provider BeforeGetRecords
event handler and a RemoteDataModule UpdateRegistry method.  The
complete project using this code example is in
Demos/DelphiWin32/VCLWin32/MIDAS/Pooler.
}
procedure TPooledRDM.DataSetProvider1BeforeGetRecords(Sender: TObject;
  var OwnerData: OleVariant);
begin
  try
    Query1.Close;
    Query1.SQL.Text := OwnerData[0];
    if not VarIsNull(OwnerData[1]) and not VarIsEmpty(OwnerData[1]) then
    begin
      Query1.Open;
      if not Query1.Locate(Query1.Fields[0].FieldName, OwnerData[1], []) then
        raise Exception.Create('Record not found');
      Query1.Next;
    end;
  finally
    OwnerData := NULL;
  end;
end;
```

See Also

AS_GetRecords

GetRecords (see page 3109)

BeforeGetRecords

AfterGetRecords (see page 3118)

Communicating with the Client Dataset

Responding to Client Data Requests

58.1.4.6.10 TCustomProvider.BeforeRowRequest Event

Occurs before the provider creates a delta packet with information about the current record.

Pascal

```
property BeforeRowRequest: TRemoteEvent;
```

C++

```
__property TRemoteEvent BeforeRowRequest;
```

Description

Write a BeforeRowRequest event handler to respond to custom information from a client dataset. BeforeRowRequest is part of the mechanism by which a provider and a client dataset communicate information about data fetching. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset fetches detail information about the current record or refreshes the current record, the following events occur:

1.The client dataset receives a BeforeRowRequest event, where it can encode custom information into an OleVariant.

2.The provider receives a BeforeRowRequest event, where the custom information appears as the OwnerData (see page 3115) parameter. The provider can respond to or change that information before it looks up the requested information.

3.The provider creates a delta packet that contains the requested information.

4.The provider receives an AfterRowRequest (see page 3119) event, where it can encode custom information into its OwnerData (see page 3115) parameter or respond to information from the BeforeRowRequest event handler.

5.The client dataset receives an AfterRowRequest (see page 3119) event, where it can respond to the custom information returned by the provider in its AfterRowRequest (see page 3119) event handler.

See Also

AS_RowRequest

RowRequest (see page 3113)

AfterRowRequest (see page 3119)

BeforeRowRequest

FetchBlobs

FetchParams

RefreshRecord

Communicating with the Client Dataset

58.1.4.6.11 TCustomProvider.OnDataRequest Event

Occurs when a provider calls its DataRequest (see page 3101) method.

Pascal

```
property OnDataRequest: TDataRequestEvent;
```

C++

```
__property TDataRequestEvent OnDataRequest;
```

Description

Write an OnDataRequest event handler to do custom processing of data. OnDataRequest has no built-in use. Instead, it allows developers to augment the IAppServer interface so that clients in a multi-tiered application can communicate directly with an individual provider.

If an application does not provide an OnDataRequest event handler, calls to DataRequest (see page 3101) return NULL.

Note: To provide the same sort of custom processing on the client side, use an automation method on the connection component.

See Also

DataRequest (see page 3101)

IAppServer

Communicating with Providers Using Custom Events

Communicating with the Client Dataset

Responding to Client-generated Events

58.1.4.6.12 TCustomProvider.OnValidate Event

Pascal

```
property OnValidate: TOnValidate;
```

C++

```
__property TOnValidate OnValidate;
```

Description

This is OnValidate, a member of class TCustomProvider.

58.1.5 TCustomResolver Class

TCustomResolver is the base class for components that apply updates on behalf of a provider.

Class Hierarchy

Pascal

```
TCustomResolver = class(TComponent);
```

C++

```
class TCustomResolver : public TComponent;
```

File

Provider

Description

TCustomResolver defines the fundamental properties and methods used by resolver components for applying updates from a client dataset or XML broker and resolving errors that occur as part of that update process.

TCustomResolver has two descendants: TDataSetResolver (see page 3176), which applies updates to a dataset, and TSQLResolver (see page 3196), which applies updates directly to a database server.

Do not create resolver instances. Provider (see page 3135) components create their own TCustomResolver descendants for applying updates.

See Also

TDataSetProvider (see page 3145)

TBaseProvider (see page 3077)

TDataSetResolver (see page 3176)

TSQLResolver (see page 3196)

58.1.5.1 TCustomResolver Members

The following tables list the members exposed by TCustomResolver.

Protected Methods

	Name	Description
	BeginUpdate (see page 3128)	Provides the interface for a method that performs any initialization before applying updates.
	DoDelete (see page 3129)	Defines a method that deletes a record.
	DoInsert (see page 3129)	Defines a method that inserts a record.
	DoUpdate (see page 3129)	Defines a method that modifies a record.
	EndUpdate (see page 3130)	Provides the interface for a method that performs any finalization after applying updates.
	HandleUpdateError (see page 3131)	Manages the resolving process when errors occur while applying updates.
	InitializeConflictBuffer (see page 3131)	Introduces a method for adding information about a failed update that is passed to an OnUpdateError event handler.
	LogUpdateError (see page 3133)	Adds an update that could not be resolved to the error log.
	LogUpdateRecord (see page 3134)	Adds records to the error log if their data has been changed by another application.

Public Methods

	Name	Description
	ApplyUpdates (see page 3128)	Applies updates received from a provider.
	Create (see page 3126)	Creates and initializes a resolver.
	Destroy (see page 3127)	Frees an instance of a resolver object.
	FreeTreeData (see page 3130)	Frees the internal data maintained by a TUpdateTree (see page 3205).
	InitKeyFields (see page 3132)	Sets the provider flag pfInKey on all key fields in a delta data packet.
	InitTreeData (see page 3132)	Initializes the internal data maintained by TUpdateTree (see page 3205).
	InternalBeforeResolve (see page 3133)	Initializes internal data structures once the delta packet has been wrapped in an update tree but before any updates are applied.
	InternalUpdateRecord (see page 3133)	Manages the updating process once a delta packet is wrapped in a TUpdateTree (see page 3205) object.
	RowRequest (see page 3134)	Returns information from a specified record.

Public Properties

	Name	Description
	Provider (see page 3135)	Indicates the provider that uses this resolver to apply updates.
	UpdateTree (see page 3135)	This is UpdateTree, a member of class TCustomResolver.

58.1.5.2 TCustomResolver.Create Constructor

Creates and initializes a resolver.

Pascal

```
constructor Create(AProvider: TBaseProvider); reintroduce;
```

C++

```
__fastcall TCustomResolver(TBaseProvider AProvider);
```

Description

Provider (see page 3135) components call the TCustomResolver (see page 3125) constructor when they need a resolver component for applying updates. The AProvider parameter specifies the provider that creates and uses the resolver.

Do not create resolver instances directly. Resolvers are designed to be created and used by providers, and can not function

independently of a provider.

See Also

Resolver (see page 3088)

58.1.5.3 TCustomResolver.Destroy Destructor

Frees an instance of a resolver object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomResolver();
```

Description

Applications rarely have to destroy resolvers explicitly. Once a resolver has been assigned as the Resolver property of a provider, the provider is responsible for freeing the resolver.

See Also

System::TObject::Free

Resolver (see page 3088)

58.1.5.4 TCustomResolver Methods

The methods of the TCustomResolver class are listed here.

Protected Methods

	Name	Description
	BeginUpdate (see page 3128)	Provides the interface for a method that performs any initialization before applying updates.
	DoDelete (see page 3129)	Defines a method that deletes a record.
	DoInsert (see page 3129)	Defines a method that inserts a record.
	DoUpdate (see page 3129)	Defines a method that modifies a record.
	EndUpdate (see page 3130)	Provides the interface for a method that performs any finalization after applying updates.
	HandleUpdateError (see page 3131)	Manages the resolving process when errors occur while applying updates.
	InitializeConflictBuffer (see page 3131)	Introduces a method for adding information about a failed update that is passed to an OnUpdateError event handler.
	LogUpdateError (see page 3133)	Adds an update that could not be resolved to the error log.
	LogUpdateRecord (see page 3134)	Adds records to the error log if their data has been changed by another application.

Public Methods

	Name	Description
	ApplyUpdates (see page 3128)	Applies updates received from a provider.
	FreeTreeData (see page 3130)	Frees the internal data maintained by a TUpdateTree (see page 3205).
	InitKeyFields (see page 3132)	Sets the provider flag pfInKey on all key fields in a delta data packet.
	InitTreeData (see page 3132)	Initializes the internal data maintained by TUpdateTree (see page 3205).
	InternalBeforeResolve (see page 3133)	Initializes internal data structures once the delta packet has been wrapped in an update tree but before any updates are applied.
	InternalUpdateRecord (see page 3133)	Manages the updating process once a delta packet is wrapped in a TUpdateTree (see page 3205) object.

	RowRequest (see page 3134)	Returns information from a specified record.

58.1.5.4.1 TCustomResolver.ApplyUpdates Method

Applies updates received from a provider.

Pascal

```
function ApplyUpdates(const Delta: OleVariant; MaxErrors: Integer; out ErrorCount: Integer):
OleVariant; virtual;
```

C++

```
virtual __fastcall OleVariant ApplyUpdates(const OleVariant Delta, int MaxErrors, int
ErrorCount);
```

Description

Providers call ApplyUpdates to apply updates received from a client dataset or XML broker.

ApplyUpdates generates an OnUpdateData event on the associated provider, and then applies the updates passed in as the Delta parameter, returning an OleVariant that contains any updates that could not be applied.

The Delta parameter specifies a delta packet that contains the insertions, deletions, and modifications to be applied. MaxErrors indicates the maximum number of errors to permit before aborting the update operation. ErrorCount returns the actual number of errors encountered during the update operation. If ErrorCount is greater than MaxErrors, (or an OnUpdateError event returns rrAbort), all updates that have been successfully applied so far are rolled back, and the update process is aborted.

If the associated provider has an OnUpdateError event handler, it is called automatically whenever a record in Delta cannot be updated. In the OnUpdateError handler, applications can correct error conditions and try reapplying an update before returning unsuccessfully updated records back to the client dataset or XML broker.

See Also

OnUpdateData (see page 3092)

OnUpdateError (see page 3092)

ApplyUpdates (see page 3100)

58.1.5.4.2 TCustomResolver.BeginUpdate Method

Provides the interface for a method that performs any initialization before applying updates.

Pascal

```
procedure BeginUpdate; virtual;
```

C++

```
virtual __fastcall BeginUpdate();
```

Description

The ApplyUpdates (see page 3128) method calls BeginUpdate before applying updates in a delta packet.

As implemented in TCustomResolver (see page 3125), BeginUpdate does nothing. Descendants that need take specific action before applying updates override this method.

See Also

EndUpdate (see page 3130)

ApplyUpdates (see page 3128)

58.1.5.4.3 TCustomResolver.DoDelete Method

Defines a method that deletes a record.

Pascal

```
procedure DoDelete(Tree: TUpdateTree); virtual; abstract;
```

C++

```
virtual __fastcall DoDelete(TUpdateTree Tree) = 0;
```

Description

DoDelete is one of several helper methods internal to the ApplyUpdates (see page 3128) method. These helper methods perform the actual updates in the delta packet. ApplyUpdates (see page 3128) wraps the delta packet in a TUpdateTree (see page 3205) object. For each record that must be deleted, the resolver calls DoDelete.

TCustomResolver (see page 3125) does not implement the DoDelete method. Descendants override this method to delete records in their own specialized way.

See Also

ApplyUpdates (see page 3128)

DoInsert (see page 3129)

DoUpdate (see page 3129)

58.1.5.4.4 TCustomResolver.DoInsert Method

Defines a method that inserts a record.

Pascal

```
procedure DoInsert(Tree: TUpdateTree); virtual; abstract;
```

C++

```
virtual __fastcall DoInsert(TUpdateTree Tree) = 0;
```

Description

DoInsert is one of several helper methods internal to the ApplyUpdates (see page 3128) method. These helper methods perform the actual updates in the delta packet. ApplyUpdates (see page 3128) wraps the delta packet in a TUpdateTree (see page 3205) object. For each record that must be inserted, the resolver calls DoInsert.

TCustomResolver (see page 3125) does not implement the DoInsert method. Descendants override this method to insert records in their own specialized way.

See Also

ApplyUpdates (see page 3128)

DoDelete (see page 3129)

DoUpdate (see page 3129)

58.1.5.4.5 TCustomResolver.DoUpdate Method

Defines a method that modifies a record.

Pascal

```
procedure DoUpdate(Tree: TUpdateTree); virtual; abstract;
```

C++

```
virtual __fastcall DoUpdate(TUpdateTree Tree) = 0;
```

Description

DoUpdate is one of several helper methods internal to the ApplyUpdates (see page 3128) method. These helper methods perform the actual updates passed in the delta packet. ApplyUpdates (see page 3128) wraps the delta packet in a TUpdateTree (see page 3205) object. For each record that must be modified, the resolver calls DoUpdate.

TCustomResolver (see page 3125) does not implement the DoUpdate method. Descendants override this method to insert records in their own specialized way.

See Also

ApplyUpdates (see page 3128)

DoDelete (see page 3129)

DoInsert (see page 3129)

58.1.5.4.6 TCustomResolver.EndUpdate Method

Provides the interface for a method that performs any finalization after applying updates.

Pascal

```
procedure EndUpdate; virtual;
```

C++

```
virtual __fastcall EndUpdate();
```

Description

The ApplyUpdates (see page 3128) method calls EndUpdate after applying updates.

As implemented in TCustomResolver (see page 3125), EndUpdate does nothing. Descendants that need take specific action after applying updates override this method.

See Also

BeginUpdate (see page 3128)

ApplyUpdates (see page 3128)

58.1.5.4.7 TCustomResolver.FreeTreeData Method

Frees the internal data maintained by a TUpdateTree (see page 3205).

Pascal

```
procedure FreeTreeData(Tree: TUpdateTree); virtual;
```

C++

```
virtual __fastcall FreeTreeData(TUpdateTree Tree);
```

Description

The ApplyUpdates (see page 3128) method wraps delta packets in a TUpdateTree (see page 3205) object. The TUpdateTree (see page 3205) stores the delta packet, along with information that the resolver uses when applying updates.

This information varies, depending on the class of resolver.

FreeTreeData is called internally just before the resolver's TUpdateTree (see page 3205) object is freed. As implemented in TCustomResolver (see page 3125), FreeTreeData does nothing. Descendants override this method if any specific action is required to free the resolver-specific information that is stored in the update tree.

See Also

ApplyUpdates (see page 3128)

InitTreeData (see page 3132)

58.1.5.4.8 TCustomResolver.HandleUpdateError Method

Manages the resolving process when errors occur while applying updates.

Pascal

```
function HandleUpdateError(Tree: TUpdateTree; E: EUpdateError; var MaxErrors: Integer; var
ErrorCount: Integer): Boolean; virtual;
```

C++

```
virtual __fastcall Boolean HandleUpdateError(TUpdateTree Tree, EUpdateError E, int MaxErrors,
int ErrorCount);
```

Description

When a resolver encounters errors in applying updates, it calls HandleUpdateError to determine how to proceed. HandleUpdateError generates an OnUpdateError event on the associated provider so that applications have an opportunity to correct update errors. Depending on the response provided by the OnUpdateError handler (if any), HandleUpdateError logs the unresolved error or signals that the resolver should try to apply the (corrected) record again.

The Tree parameter specifies the TUpdateTree (see page 3205) that wraps all the updates currently being applied. The MaxErrors parameter indicates the maximum number of errors that the resolver will tolerate before the update operation is rolled back. The ErrorCount parameter specifies how many errors have been encountered so far.

HandleUpdateError returns true if the record has been corrected and the resolver should try to reapply the update. It returns false if the record is added to the error log.

See Also

OnUpdateError (see page 3092)

ApplyUpdates (see page 3128)

58.1.5.4.9 TCustomResolver.InitializeConflictBuffer Method

Introduces a method for adding information about a failed update that is passed to an OnUpdateError event handler.

Pascal

```
procedure InitializeConflictBuffer(Tree: TUpdateTree); virtual; abstract;
```

C++

```
virtual __fastcall InitializeConflictBuffer(TUpdateTree Tree) = 0;
```

Description

The HandleUpdateError (see page 3131) method calls InitializeConflictBuffer before generating an OnUpdateError event. This ensures that the client dataset passed to an OnUpdateError event handler represents the update that caused the problem, with appropriate values for field's CurValue property.

Descendants of TCustomResolver (see page 3125) implement this method to obtain information from the dataset or server to which they are applying updates.

See Also

HandleUpdateError (see page 3131)

OnUpdateError (see page 3092)

58.1.5.4.10 TCustomResolver.InitKeyFields Method

Sets the provider flag pfInKey on all key fields in a delta data packet.

Pascal

```
procedure InitKeyFields(Tree: TUpdateTree; ADelta: TPacketDataSet); virtual;
```

C++

```
virtual __fastcall InitKeyFields(TUpdateTree Tree, TPacketDataSet ADelta);
```

Description

InitKeyFields is used internally to mark fields from a delta packet that can be used to uniquely identify records in the underlying dataset.

The Tree parameter is the internal TUpdateTree (see page 3205) that the resolver uses to store information about the delta packet. The ADelta parameter is the delta packet of changes to apply. For every field in this delta packet that acts as a key value to uniquely identify records, InitKeyFields adds pfInKey to its ProviderFlags property.

See Also

ProviderFlags

58.1.5.4.11 TCustomResolver.InitTreeData Method

Initializes the internal data maintained by TUpdateTree (see page 3205).

Pascal

```
procedure InitTreeData(Tree: TUpdateTree); virtual;
```

C++

```
virtual __fastcall InitTreeData(TUpdateTree Tree);
```

Description

The ApplyUpdates (see page 3128) method wraps delta packets in a TUpdateTree (see page 3205) object. The TUpdateTree (see page 3205) stores the delta packet, along with information that the resolver uses when applying updates. This information varies, depending on the class of resolver.

InitTreeData is called internally when the resolver's TUpdateTree (see page 3205) object is created. As implemented in TCustomResolver (see page 3125), InitTreeData does nothing. Descendants override this method to store any class-specific information that they use when applying updates.

See Also

ApplyUpdates (see page 3128)

FreeTreeData (see page 3130)

58.1.5.4.12 TCustomResolver.InternalBeforeResolve Method

Initializes internal data structures once the delta packet has been wrapped in an update tree but before any updates are applied.

Pascal

```
procedure InternalBeforeResolve(Tree: TUpdateTree); virtual;
```

C++

```
virtual __fastcall InternalBeforeResolve(TUpdateTree Tree);
```

Description

The ApplyUpdates (see page 3128) method wraps delta packets in a TUpdateTree (see page 3205) object. Once this is complete, InternalBeforeResolve performs any other initialization before the updates are applied.

As implemented in TCustomResolver (see page 3125), InternalBeforeResolve does nothing. Descendants override this method to perform class-specific initialization.

See Also

ApplyUpdates (see page 3128)

InitTreeData (see page 3132)

58.1.5.4.13 TCustomResolver.InternalUpdateRecord Method

Manages the updating process once a delta packet is wrapped in a TUpdateTree (see page 3205) object.

Pascal

```
function InternalUpdateRecord(Tree: TUpdateTree): Boolean; virtual;
```

C++

```
virtual __fastcall Boolean InternalUpdateRecord(TUpdateTree Tree);
```

Description

InternalUpdateRecord orchestrates the updating process, ensuring that updates are applied in the correct order and that errors are resolved or logged. For each record that must be updated, InternalUpdateRecord

1.Generates a BeforeUpdateRecord event.

2.Applies the update.

3.Generates an AfterUpdateRecord event if the update is successful, or calls HandleUpdateError (see page 3131) to generate an OnUpdateError event.

See Also

ApplyUpdates (see page 3128)

BeforeUpdateRecord (see page 3090)

AfterUpdateRecord (see page 3090)

OnUpdateError (see page 3092)

HandleUpdateError (see page 3131)

58.1.5.4.14 TCustomResolver.LogUpdateError Method

Adds an update that could not be resolved to the error log.

Pascal

```
procedure LogUpdateError(Tree: TUpdateTree; E: EUpdateError; Response: TResolverResponse);
virtual;
```

C++

```
virtual __fastcall LogUpdateError(TUpdateTree Tree, EUpdateError E, TResolverResponse
Response);
```

Description

When the HandleUpdateError (see page 3131) method can't resolve an update conflict, it calls LogUpdateError to add the unapplied update to the error log. This error log is eventually returned by the ApplyUpdates (see page 3128) method when the update operation ends.

The Tree parameter represents the set of updates in the delta packet. The E parameter is the EUpdateError exception that represents the current update error. The Response parameter is rrSkip if the update process should continue after the update is logged, and rrAbort if the entire update operation should be rolled back. LogUpdateError is never called with Response set to any other value.

See Also

HandleUpdateError (see page 3131)

OnUpdateError (see page 3092)

58.1.5.4.15 TCustomResolver.LogUpdateRecord Method

Adds records to the error log if their data has been changed by another application.

Pascal

```
procedure LogUpdateRecord(Tree: TUpdateTree); virtual;
```

C++

```
virtual __fastcall LogUpdateRecord(TUpdateTree Tree);
```

Description

When the InternalUpdateRecord (see page 3133) method detects that a record includes other changes than those specified by an update in the delta packet, it calls LogUpdateRecord to add the changed and updated record to the error log so that the client dataset or XML broker can refresh the updated record. This error log is eventually returned by the ApplyUpdates (see page 3128) method when the update operation ends.

The Tree parameter represents the set of updates being applied.

See Also

LogUpdateError (see page 3133)

InternalUpdateRecord (see page 3133)

58.1.5.4.16 TCustomResolver.RowRequest Method

Returns information from a specified record.

Pascal

```
function RowRequest(Row: OleVariant; Options: TFetchOptions): OleVariant; virtual;
```

C++

```
virtual __fastcall OleVariant RowRequest(OleVariant Row, TFetchOptions Options);
```

Description

RowRequest is called automatically in response to the associated provider's RowRequest method.

The Row parameter is an OleVariant that describes the record about which information is requested. The resolver uses this to locate the desired record.

The requested information is returned as a delta packet.

See Also

RowRequest (see page 3113)

58.1.5.5 TCustomResolver Properties

The properties of the TCustomResolver class are listed here.

Public Properties

	Name	Description
	Provider (see page 3135)	Indicates the provider that uses this resolver to apply updates.
	UpdateTree (see page 3135)	This is UpdateTree, a member of class TCustomResolver.

58.1.5.5.1 TCustomResolver.Provider Property

Indicates the provider that uses this resolver to apply updates.

Pascal

```
property Provider: TBaseProvider;
```

C++

```
__property TBaseProvider Provider;
```

Description

Provider is a read-only property that points to the provider that uses this resolver for applying updates. The Provider property is set in the constructor.

Resolvers use the Provider property internally to obtain information supplied in the provider's properties. They also use it to generate events on the provider at appropriate points in the update process.

See Also

Create (see page 3126)

Resolver (see page 3088)

58.1.5.5.2 TCustomResolver.UpdateTree Property

Pascal

```
property UpdateTree: TUpdateTree;
```

C++

```
__property TUpdateTree UpdateTree;
```

Description

This is UpdateTree, a member of class TCustomResolver.

58.1.6 TDataPacketWriter Class

Class Hierarchy

Pascal

```
TDataPacketWriter = class(TCustomPacketWriter);
```

C++

```
class TDataPacketWriter : public TCustomPacketWriter;
```

File

Provider

Description

This is class Provider.TDataPacketWriter.

58.1.6.1 TDataPacketWriter Members

The following tables list the members exposed by TDataPacketWriter.

Protected Fields

	Name	Description
	FBuffer (see page 3095)	This is FBuffer, a member of class TCustomPacketWriter.

Protected Methods

	Name	Description
V	AddAttribute (see page 3095)	This is AddAttribute, a member of class TCustomPacketWriter.
V	AddColumn (see page 3138)	This is AddColumn, a member of class TDataPacketWriter.
V	AddConstraints (see page 3138)	This is AddConstraints, a member of class TDataPacketWriter.
V	AddDataSetAttributes (see page 3139)	This is AddDataSetAttributes, a member of class TDataPacketWriter.
	AddExtraFieldProps (see page 3139)	This is AddExtraFieldProps, a member of class TDataPacketWriter.
V	AddFieldLinks (see page 3139)	This is AddFieldLinks, a member of class TDataPacketWriter.
V	AddIndexDefs (see page 3139)	This is AddIndexDefs, a member of class TDataPacketWriter.
	Check (see page 3095)	This is Check, a member of class TCustomPacketWriter.
	FreeInfoRecords (see page 3140)	This is FreeInfoRecords, a member of class TDataPacketWriter.
	GetFieldIdx (see page 3140)	This is GetFieldIdx, a member of class TDataPacketWriter.
	InitPutProcs (see page 3140)	This is InitPutProcs, a member of class TDataPacketWriter.
	PutADTField (see page 3140)	This is PutADTField, a member of class TDataPacketWriter.
	PutArrayField (see page 3141)	This is PutArrayField, a member of class TDataPacketWriter.
	PutBlobField (see page 3141)	This is PutBlobField, a member of class TDataPacketWriter.
	PutCalcField (see page 3141)	This is PutCalcField, a member of class TDataPacketWriter.
	PutDataSetField (see page 3141)	This is PutDataSetField, a member of class TDataPacketWriter.
	PutField (see page 3142)	This is PutField, a member of class TDataPacketWriter.
	PutStringField (see page 3142)	This is PutStringField, a member of class TDataPacketWriter.
	PutVarBytesField (see page 3142)	This is PutVarBytesField, a member of class TDataPacketWriter.
	PutWideStringField (see page 3142)	This is PutWideStringField, a member of class TDataPacketWriter.
	RefreshPutProcs (see page 3142)	This is RefreshPutProcs, a member of class TDataPacketWriter.
	SetDSWriter (see page 3096)	This is SetDSWriter, a member of class TCustomPacketWriter.
V	WriteDataSet (see page 3143)	This is WriteDataSet, a member of class TDataPacketWriter.

	WriteMetaData (see page 3143)	This is WriteMetaData, a member of class TDataPacketWriter.

Protected Properties

	Name	Description
	DSWriter (see page 3096)	This is DSWriter, a member of class TCustomPacketWriter.

Public Events

	Name	Description
	OnGetParams (see page 3144)	This is OnGetParams, a member of class TDataPacketWriter.

Public Methods

	Name	Description
	Create (see page 3094)	Constructs an object and initializes its data before the object is first used.
	Destroy (see page 3137)	Disposes of an object instance.
	GetDataPacket (see page 3140)	This is GetDataPacket, a member of class TDataPacketWriter.
	Reset (see page 3143)	This is Reset, a member of class TDataPacketWriter.

Public Properties

	Name	Description
	Constraints (see page 3144)	This is Constraints, a member of class TDataPacketWriter.
	Options (see page 3144)	This is Options, a member of class TDataPacketWriter.
	PacketOptions (see page 3144)	This is PacketOptions, a member of class TDataPacketWriter.

58.1.6.2 TDataPacketWriter.Destroy Destructor

Disposes of an object instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TDataPacketWriter();
```

Description

Do not call System::TObject::Destroy directly. Call System::TObject::Free instead. Free verifies that the object reference is not nil before calling System::TObject::Destroy.

The System::TObject::Destroy method defined by System::TObject deallocates memory. Descendant objects usually define a destructor that is customized for that particular kind of object.

When declaring a System::TObject::Destroy method in a descendant, always add the override directive to the declaration and call the inherited System::TObject::Destroy as the last statement in the overriding method. Since System::TObject::Destroy is a virtual method, overriding it ensures that the proper inherited behavior occurs.

Note: If an exception escapes from the constructor, the destructor is called to destroy the partially constructed object instance that failed to initialize completely. Therefore, destructors should check that allocated resources, such as handles, were actually allocated before trying to release them, since their value might be zero.

System::TObject::Destroy should be implemented so that it calls System::TObject::Free on all subobjects created within the object's constructor (that is, allocated by the constructor). Unlike System::TObject::Destroy, System::TObject::Free provides a safeguard when destroying objects that are nil.

See Also

System::TObject::Free

58.1.6.3 TDataPacketWriter Methods

The methods of the TDataPacketWriter class are listed here.

Protected Methods

	Name	Description
	AddColumn (see page 3138)	This is AddColumn, a member of class TDataPacketWriter.
	AddConstraints (see page 3138)	This is AddConstraints, a member of class TDataPacketWriter.
	AddDataSetAttributes (see page 3139)	This is AddDataSetAttributes, a member of class TDataPacketWriter.
	AddExtraFieldProps (see page 3139)	This is AddExtraFieldProps, a member of class TDataPacketWriter.
	AddFieldLinks (see page 3139)	This is AddFieldLinks, a member of class TDataPacketWriter.
	AddIndexDefs (see page 3139)	This is AddIndexDefs, a member of class TDataPacketWriter.
	FreeInfoRecords (see page 3140)	This is FreeInfoRecords, a member of class TDataPacketWriter.
	GetFieldIdx (see page 3140)	This is GetFieldIdx, a member of class TDataPacketWriter.
	InitPutProcs (see page 3140)	This is InitPutProcs, a member of class TDataPacketWriter.
	PutADTField (see page 3140)	This is PutADTField, a member of class TDataPacketWriter.
	PutArrayField (see page 3141)	This is PutArrayField, a member of class TDataPacketWriter.
	PutBlobField (see page 3141)	This is PutBlobField, a member of class TDataPacketWriter.
	PutCalcField (see page 3141)	This is PutCalcField, a member of class TDataPacketWriter.
	PutDataSetField (see page 3141)	This is PutDataSetField, a member of class TDataPacketWriter.
	PutField (see page 3142)	This is PutField, a member of class TDataPacketWriter.
	PutStringField (see page 3142)	This is PutStringField, a member of class TDataPacketWriter.
	PutVarBytesField (see page 3142)	This is PutVarBytesField, a member of class TDataPacketWriter.
	PutWideStringField (see page 3142)	This is PutWideStringField, a member of class TDataPacketWriter.
	RefreshPutProcs (see page 3142)	This is RefreshPutProcs, a member of class TDataPacketWriter.
	WriteDataSet (see page 3143)	This is WriteDataSet, a member of class TDataPacketWriter.
	WriteMetaData (see page 3143)	This is WriteMetaData, a member of class TDataPacketWriter.

Public Methods

	Name	Description
	GetDataPacket (see page 3140)	This is GetDataPacket, a member of class TDataPacketWriter.
	Reset (see page 3143)	This is Reset, a member of class TDataPacketWriter.

58.1.6.3.1 TDataPacketWriter.AddColumn Method

Pascal

```
procedure AddColumn(const Info: TPutFieldInfo); virtual;
```

C++

```
virtual __fastcall AddColumn(const TPutFieldInfo Info);
```

Description

This is AddColumn, a member of class TDataPacketWriter.

58.1.6.3.2 TDataPacketWriter.AddConstraints Method

Pascal

```
procedure AddConstraints(DataSet: TDataSet); virtual;
```

C++

```
virtual __fastcall AddConstraints(TDataSet DataSet);
```

Description

This is AddConstraints, a member of class TDataPacketWriter.

58.1.6.3.3 TDataPacketWriter.AddDataSetAttributes Method

Pascal

```
procedure AddDataSetAttributes(DataSet: TDataSet); virtual;
```

C++

```
virtual __fastcall AddDataSetAttributes(TDataSet DataSet);
```

Description

This is AddDataSetAttributes, a member of class TDataPacketWriter.

58.1.6.3.4 TDataPacketWriter.AddExtraFieldProps Method

Pascal

```
procedure AddExtraFieldProps(Field: TField);
```

C++

```
__fastcall AddExtraFieldProps(TField Field);
```

Description

This is AddExtraFieldProps, a member of class TDataPacketWriter.

58.1.6.3.5 TDataPacketWriter.AddFieldLinks Method

Pascal

```
procedure AddFieldLinks(const Info: TInfoArray); virtual;
```

C++

```
virtual __fastcall AddFieldLinks(const TInfoArray Info);
```

Description

This is AddFieldLinks, a member of class TDataPacketWriter.

58.1.6.3.6 TDataPacketWriter.AddIndexDefs Method

Pascal

```
procedure AddIndexDefs(DataSet: TDataSet; const Info: TInfoArray); virtual;
```

C++

```
virtual __fastcall AddIndexDefs(TDataSet DataSet, const TInfoArray Info);
```

Description

This is AddIndexDefs, a member of class TDataPacketWriter.

58.1.6.3.7 TDataPacketWriter.FreeInfoRecords Method

Pascal

```
procedure FreeInfoRecords(var Info: TInfoArray);
```

C++

```
__fastcall FreeInfoRecords(TInfoArray Info);
```

Description

This is FreeInfoRecords, a member of class TDataPacketWriter.

58.1.6.3.8 TDataPacketWriter.GetDataPacket Method

Pascal

```
procedure GetDataPacket(DataSet: TDataSet; var RecsOut: Integer; out Data: OleVariant);
virtual;
```

C++

```
virtual __fastcall GetDataPacket(TDataSet DataSet, int RecsOut, OleVariant Data);
```

Description

This is GetDataPacket, a member of class TDataPacketWriter.

58.1.6.3.9 TDataPacketWriter.GetFieldIdx Method

Pascal

```
function GetFieldIdx(const FieldName: WideString; const Info: TInfoArray): Integer;
```

C++

```
__fastcall int GetFieldIdx(const BSTR FieldName, const TInfoArray Info);
```

Description

This is GetFieldIdx, a member of class TDataPacketWriter.

58.1.6.3.10 TDataPacketWriter.InitPutProcs Method

Pascal

```
function InitPutProcs(ADataSet: TDataSet; var GlobalIdx: Integer): TInfoArray;
```

C++

```
__fastcall TInfoArray InitPutProcs(TDataSet ADataSet, int GlobalIdx);
```

Description

This is InitPutProcs, a member of class TDataPacketWriter.

58.1.6.3.11 TDataPacketWriter.PutADTField Method

Pascal

```
procedure PutADTField(Info: PPutFieldInfo);
```

C++

```
__fastcall PutADTField(PPutFieldInfo Info);
```

Description

This is PutADTField, a member of class TDataPacketWriter.

58.1.6.3.12 TDataPacketWriter.PutArrayField Method

Pascal

```
procedure PutArrayField(Info: PPutFieldInfo);
```

C++

```
__fastcall PutArrayField(PPutFieldInfo Info);
```

Description

This is PutArrayField, a member of class TDataPacketWriter.

58.1.6.3.13 TDataPacketWriter.PutBlobField Method

Pascal

```
procedure PutBlobField(Info: PPutFieldInfo);
```

C++

```
__fastcall PutBlobField(PPutFieldInfo Info);
```

Description

This is PutBlobField, a member of class TDataPacketWriter.

58.1.6.3.14 TDataPacketWriter.PutCalcField Method

Pascal

```
procedure PutCalcField(Info: PPutFieldInfo);
```

C++

```
__fastcall PutCalcField(PPutFieldInfo Info);
```

Description

This is PutCalcField, a member of class TDataPacketWriter.

58.1.6.3.15 TDataPacketWriter.PutDataSetField Method

Pascal

```
procedure PutDataSetField(Info: PPutFieldInfo);
```

C++

```
__fastcall PutDataSetField(PPutFieldInfo Info);
```

Description

This is PutDataSetField, a member of class TDataPacketWriter.

58.1.6.3.16 TDataPacketWriter.PutField Method

Pascal

```
procedure PutField(Info: PPutFieldInfo);
```

C++

```
__fastcall PutField(PPutFieldInfo Info);
```

Description

This is PutField, a member of class TDataPacketWriter.

58.1.6.3.17 TDataPacketWriter.PutStringField Method

Pascal

```
procedure PutStringField(Info: PPutFieldInfo);
```

C++

```
__fastcall PutStringField(PPutFieldInfo Info);
```

Description

This is PutStringField, a member of class TDataPacketWriter.

58.1.6.3.18 TDataPacketWriter.PutVarBytesField Method

Pascal

```
procedure PutVarBytesField(Info: PPutFieldInfo);
```

C++

```
__fastcall PutVarBytesField(PPutFieldInfo Info);
```

Description

This is PutVarBytesField, a member of class TDataPacketWriter.

58.1.6.3.19 TDataPacketWriter.PutWideStringField Method

Pascal

```
procedure PutWideStringField(Info: PPutFieldInfo);
```

C++

```
__fastcall PutWideStringField(PPutFieldInfo Info);
```

Description

This is PutWideStringField, a member of class TDataPacketWriter.

58.1.6.3.20 TDataPacketWriter.RefreshPutProcs Method

Pascal

```
procedure RefreshPutProcs(ADataSet: TDataSet; var Info: TInfoArray);
```

C++

```
__fastcall RefreshPutProcs(TDataSet ADataSet, TInfoArray Info);
```

Description

This is RefreshPutProcs, a member of class TDataPacketWriter.

58.1.6.3.21 TDataPacketWriter.Reset Method

Pascal

```
procedure Reset;
```

C++

```
__fastcall Reset();
```

Description

This is Reset, a member of class TDataPacketWriter.

58.1.6.3.22 TDataPacketWriter.WriteDataSet Method

Pascal

```
function WriteDataSet(DataSet: TDataSet; var Info: TInfoArray; RecsOut: Integer): Integer;
virtual;
```

C++

```
virtual __fastcall int WriteDataSet(TDataSet DataSet, TInfoArray Info, int RecsOut);
```

Description

This is WriteDataSet, a member of class TDataPacketWriter.

58.1.6.3.23 TDataPacketWriter.WriteMetaData Method

Pascal

```
procedure WriteMetaData(DataSet: TDataSet; const Info: TInfoArray; IsReference: Boolean =
False); virtual;
```

C++

```
virtual __fastcall WriteMetaData(TDataSet DataSet, const TInfoArray Info, Boolean IsReference
= False);
```

Description

This is WriteMetaData, a member of class TDataPacketWriter.

58.1.6.4 TDataPacketWriter Properties

The properties of the TDataPacketWriter class are listed here.

Public Properties

	Name	Description
	Constraints (see page 3144)	This is Constraints, a member of class TDataPacketWriter.
	Options (see page 3144)	This is Options, a member of class TDataPacketWriter.
	PacketOptions (see page 3144)	This is PacketOptions, a member of class TDataPacketWriter.

58.1.6.4.1 TDataPacketWriter.Constraints Property

Pascal

```
property Constraints: Boolean;
```

C++

```
__property Boolean Constraints;
```

Description

This is Constraints, a member of class TDataPacketWriter.

58.1.6.4.2 TDataPacketWriter.Options Property

Pascal

```
property Options: TProviderOptions;
```

C++

```
__property TProviderOptions Options;
```

Description

This is Options, a member of class TDataPacketWriter.

58.1.6.4.3 TDataPacketWriter.PacketOptions Property

Pascal

```
property PacketOptions: TGetRecordOptions;
```

C++

```
__property TGetRecordOptions PacketOptions;
```

Description

This is PacketOptions, a member of class TDataPacketWriter.

58.1.6.5 TDataPacketWriter Events

The events of the TDataPacketWriter class are listed here.

Public Events

	Name	Description
	OnGetParams (see page 3144)	This is OnGetParams, a member of class TDataPacketWriter.

58.1.6.5.1 TDataPacketWriter.OnGetParams Event

Pascal

```
property OnGetParams: TGetParamsEvent;
```

C++

```
__property TGetParamsEvent OnGetParams;
```

Description

This is OnGetParams, a member of class TDataPacketWriter.

58.1.7 TDataSetProvider Class

TDataSetProvider provides data from a dataset and resolves updates to that dataset or its database server.

Class Hierarchy

Pascal

```
TDataSetProvider = class(TBaseProvider);
```

C++

```
class TDataSetProvider : public TBaseProvider;
```

File

Provider

Description

Use TDataSetProvider to provide data from a dataset to a client dataset or XML broker and to resolve updates from that client dataset or XML broker back to the dataset or to its underlying database server. TDataSetProvider can be a part of the same application as the client dataset or XML broker, or it can be placed in the application server of a multi-tiered application. It serves as a data broker between a remote database server and a client dataset.

TDataSetProvider packages data from a dataset and passes it in one or more transportable data packets to the client dataset or XML broker. The client dataset reconstructs the data in the data packet to create a local, in-memory copy for user access. When user access is complete, the client dataset repackages any changed data and sends the updates back to the provider. The provider applies the updates to the database or source dataset.

When the provider supplies data to a client dataset, the client dataset reconstructs the data in the data packet to create a local, in-memory copy for user access. When user access is complete, the client dataset repackages any changed data and sends the updates back to the provider. The provider applies the updates to the database or source dataset.

When the provider supplies data to an XML broker, the XML broker adds the data packet, in XML format, to an HTML document that is downloaded by a Web client. When the XML broker receives updates from the Web client, it sends the updates to the provider, which applies them to the database or source dataset.

Client datasets and XML brokers communicate with a provider through the IAppServer interface. In multi-tiered applications, this is the interface of the remote data module that contains the provider. To enable the remote data module to pass calls to the provider, the provider's Exported (see page 3160) property must be true, and its Owner property must specify the remote data module.

See Also

TClientDataSet

IAppServer

Determining the Source of Data

Using Provider Components

58.1.7.1 TDataSetProvider Members

The following tables list the members exposed by TDataSetProvider.

Protected Methods

	Name	Description
	CheckDataSet (see page 3149)	This is CheckDataSet, a member of class TDataSetProvider.
	CheckResolver (see page 3081)	Instantiates a resolver for the Resolver (see page 3088) property if it does not already exist.
V	CreateDataPacket (see page 3149)	Generates a data packet for the data associated with the provider.
V	CreateResolver (see page 3150)	Creates a resolver component for handling the application of updates.
V	DoAfterApplyUpdates (see page 3102)	Generates an AfterApplyUpdates (see page 3116) event.
V	DoAfterExecute (see page 3102)	Generates an AfterExecute (see page 3117) event.
V	DoAfterGetParams (see page 3102)	Generates an AfterGetParams (see page 3118) event.
V	DoAfterGetRecords (see page 3103)	Generates an AfterGetRecords (see page 3118) event.
V	DoAfterRowRequest (see page 3103)	Generates an AfterRowRequest (see page 3119) event.
V	DoAfterUpdateRecord (see page 3082)	Generates an AfterUpdateRecord (see page 3090) event.
V	DoApplyUpdates (see page 3104)	This is DoApplyUpdates, a member of class TCustomProvider.
V	DoBeforeApplyUpdates (see page 3104)	Generates a BeforeApplyUpdates (see page 3120) event.
V	DoBeforeCommit (see page 3082)	This is DoBeforeCommit, a member of class TBaseProvider.
V	DoBeforeExecute (see page 3150)	Generates a BeforeExecute (see page 3169) event.
V	DoBeforeGetParams (see page 3105)	Generates a BeforeGetParams (see page 3121) event.
V	DoBeforeGetRecords (see page 3151)	Generates a BeforeGetRecords (see page 3170) event.
V	DoBeforeRowRequest (see page 3106)	Generates a BeforeRowRequest (see page 3123) event.
V	DoBeforeUpdateRecord (see page 3083)	This is DoBeforeUpdateRecord, a member of class TBaseProvider.
V	DoGetProviderAttributes (see page 3152)	Fills a list with information about the dataset that can be used when providing or resolving.
V	DoGetRecords (see page 3107)	This is DoGetRecords, a member of class TCustomProvider.
V	DoGetTableName (see page 3152)	Generates an OnGetTableName (see page 3175) event.
V	DoOnGetData (see page 3083)	Generates an OnGetData (see page 3091) event.
V	DoOnUpdateData (see page 3083)	Generates an OnUpdateData (see page 3092) event.
V	FetchDetails (see page 3153)	Fills a delta dataset with all nested dataset field values.
V	FindRecord (see page 3153)	This is FindRecord, a member of class TDataSetProvider.
	FreeResolver (see page 3084)	Frees the resolver specified by the Resolver (see page 3088) property.
V	GetDataSetFromDelta (see page 3153)	Returns a dataset positioned on the record of the source dataset that is currently being fetched or updated.
V	InternalApplyUpdates (see page 3154)	Provides the underlying implementation of the ApplyUpdates (see page 3100) method.
V	InternalExecute (see page 3154)	Provides the underlying implementation of the Execute method.
V	InternalGetParams (see page 3155)	Returns the requested parameter values as a Variant array.
V	InternalGetRecords (see page 3155)	Provides the underlying implementation of the GetRecords (see page 3109) method.
V	InternalRowRequest (see page 3156)	Provides the underlying implementation of the RowRequest (see page 3113) method.
V	LocateRecord (see page 3157)	Positions a data set on the record that corresponds to the current update in a specified delta packet.
V	Notification (see page 3157)	Responds to notifications that objects are being created or destroyed.
	Reset (see page 3157)	This is Reset, a member of class TDataSetProvider.
	SetActiveUpdateException (see page 3114)	This is SetActiveUpdateException, a member of class TCustomProvider.
V	SetCommandText (see page 3158)	This is SetCommandText, a member of class TDataSetProvider.
V	SetParams (see page 3158)	This is SetParams, a member of class TDataSetProvider.
V	UpdateRecord (see page 3158)	Retrieves specified data from a source dataset into a delta packet.

Protected Properties

	Name	Description
	PacketDataSet (see page 3088)	This is PacketDataSet, a member of class TBaseProvider.
	TransactionStarted (see page 3163)	This is TransactionStarted, a member of class TDataSetProvider.

Public Methods

	Name	Description
	ApplyUpdates (see page 3100)	This is the overview for the ApplyUpdates method overload.
V	Create (see page 3148)	Creates and initializes a TDataSetProvider (see page 3145) object.
V	DataRequest (see page 3101)	Generates an OnDataRequest (see page 3124) event.
V	Destroy (see page 3148)	Frees an instance of a data set provider.
V	Execute (see page 3107)	Executes a query, stored procedure, or custom SQL statement.
V	GetParams (see page 3108)	Returns the current parameter values.
	GetRecords (see page 3109)	This is the overview for the GetRecords method overload.
V	RowRequest (see page 3113)	Returns information from a specified record.

Public Properties

	Name	Description
	ActiveUpdateException (see page 3159)	This is ActiveUpdateException, a member of class TDataSetProvider.
R	Data (see page 3115)	Denotes a data packet created by the provider.
	OwnerData (see page 3162)	This is OwnerData, a member of class TDataSetProvider.
R	Params (see page 3162)	Stores the parameter values currently in use.
R	Resolver (see page 3088)	Provides access to the resolver component that applies updates and resolves update errors.

Published Events

	Name	Description
	AfterApplyUpdates (see page 3164)	Occurs after the provider applies updates.
	AfterExecute (see page 3165)	Occurs after the provider passes an execute command to its dataset.
	AfterGetParams (see page 3166)	Occurs after the provider creates an OleVariant containing parameter values to send to the client dataset.
	AfterGetRecords (see page 3166)	Occurs after the provider creates a data packet to send to the client dataset.
	AfterRowRequest (see page 3167)	Occurs after the provider fetches new information about the current record.
	AfterUpdateRecord (see page 3168)	Occurs when the associated resolver has just updated a record.
	BeforeApplyUpdates (see page 3168)	Occurs before the provider applies updates from a client dataset.
	BeforeExecute (see page 3169)	Occurs before the provider passes an execute command to its dataset.
	BeforeGetParams (see page 3170)	Occurs before the provider creates an OleVariant that encodes the current parameter values and sends them to a client dataset.
	BeforeGetRecords (see page 3170)	Occurs before the provider creates a data packet and sends it to a client dataset.
	BeforeRowRequest (see page 3172)	Occurs before the provider creates a delta packet with information about the current record.
	BeforeUpdateRecord (see page 3172)	Occurs immediately before each record is applied to the remote dataset.
	OnDataRequest (see page 3173)	Occurs when a provider calls its DataRequest method.
	OnGetData (see page 3174)	Occurs after a provider retrieves data but before it is sent to the client.
	OnGetDataSetProperties (see page 3174)	Occurs when the provider adds additional information to a data packet it is creating.
	OnGetTableName (see page 3175)	Occurs when a resolver initializes its information about the table to which it applies updates.
	OnUpdateData (see page 3175)	Occurs when a provider begins to apply updates.
	OnUpdateError (see page 3176)	Occurs when a provider can't update a record because of an error condition.
	OnValidate (see page 3176)	This is OnValidate, a member of class TDataSetProvider.

Published Properties

	Name	Description
	Constraints (see page 3159)	Controls whether database constraints are passed to the client so that it can check them locally.

	DataSet (see page 3160)	Specifies the dataset from which the provider provides data and to which it applies updates.
	Exported (see page 3160)	Allows client applications to pass calls to the provider when it resides on an application server.
	Options (see page 3161)	Influences what is included in data packets and how the information in data packets is used.
	ResolveToDataSet (see page 3162)	Specifies whether updates should be applied to a dataset or directly to the database server.
	UpdateMode (see page 3163)	Controls how the provider locates records when applying updates.

58.1.7.2 TDataSetProvider.Create Constructor

Creates and initializes a TDataSetProvider (see page 3145) object.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TDataSetProvider(TComponent * AOwner);
```

Description

Call Create to instantiate a provider at runtime. Providers placed in forms or data modules at design time are created automatically.

See Also

Create (see page 3079)

58.1.7.3 TDataSetProvider.Destroy Destructor

Frees an instance of a data set provider.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TDataSetProvider();
```

Description

Do not call Destroy directly. Instead, call Free, which checks that the reference to the provider is not nil before calling its Destroy method. Destroy frees all helper objects, including the resolver component referenced by the Resolver (see page 3088) property.

See Also

System::TObject::Free

Resolver (see page 3088)

58.1.7.4 TDataSetProvider Methods

The methods of the TDataSetProvider class are listed here.

Protected Methods

	Name	Description
	CheckDataSet (see page 3149)	This is CheckDataSet, a member of class TDataSetProvider.
V	CreateDataPacket (see page 3149)	Generates a data packet for the data associated with the provider.
V	CreateResolver (see page 3150)	Creates a resolver component for handling the application of updates.
V	DoBeforeExecute (see page 3150)	Generates a BeforeExecute (see page 3169) event.
V	DoBeforeGetRecords (see page 3151)	Generates a BeforeGetRecords (see page 3170) event.
V	DoGetProviderAttributes (see page 3152)	Fills a list with information about the dataset that can be used when providing or resolving.
V	DoGetTableName (see page 3152)	Generates an OnGetTableName (see page 3175) event.
V	FetchDetails (see page 3153)	Fills a delta dataset with all nested dataset field values.
V	FindRecord (see page 3153)	This is FindRecord, a member of class TDataSetProvider.
V	GetDataSetFromDelta (see page 3153)	Returns a dataset positioned on the record of the source dataset that is currently being fetched or updated.
V	InternalApplyUpdates (see page 3154)	Provides the underlying implementation of the ApplyUpdates (see page 3100) method.
V	InternalExecute (see page 3154)	Provides the underlying implementation of the Execute method.
V	InternalGetParams (see page 3155)	Returns the requested parameter values as a Variant array.
V	InternalGetRecords (see page 3155)	Provides the underlying implementation of the GetRecords (see page 3109) method.
V	InternalRowRequest (see page 3156)	Provides the underlying implementation of the RowRequest (see page 3113) method.
V	LocateRecord (see page 3157)	Positions a data set on the record that corresponds to the current update in a specified delta packet.
V	Notification (see page 3157)	Responds to notifications that objects are being created or destroyed.
	Reset (see page 3157)	This is Reset, a member of class TDataSetProvider.
V	SetCommandText (see page 3158)	This is SetCommandText, a member of class TDataSetProvider.
V	SetParams (see page 3158)	This is SetParams, a member of class TDataSetProvider.
V	UpdateRecord (see page 3158)	Retrieves specified data from a source dataset into a delta packet.

58.1.7.4.1 TDataSetProvider.CheckDataSet Method

Pascal

```
procedure CheckDataSet;
```

C++

```
__fastcall CheckDataSet();
```

Description

This is CheckDataSet, a member of class TDataSetProvider.

58.1.7.4.2 TDataSetProvider.CreateDataPacket Method

Generates a data packet for the data associated with the provider.

Pascal

```
procedure CreateDataPacket(PacketOpts: TGetRecordOptions; ProvOpts: TProviderOptions; var
RecsOut: Integer; var Data: OleVariant); override;
```

C++

```
virtual __fastcall CreateDataPacket(TGetRecordOptions PacketOpts, TProviderOptions ProvOpts,
int RecsOut, OleVariant Data);
```

Description

CreateDataPacket is used internally to generate the data packets that are returned by GetRecords (see page 3109).

The PacketOpts parameter indicates what information (other than record values) is included in the data packet. It reflects any Options (see page 3161) passed to the GetRecords (see page 3109) method.

The ProvOpts parameter reflects the current value of the Options (see page 3161) property.

The RecsOut parameter specifies the number of records desired in the packet and returns the actual number of records in the generated data packet.

The Data (see page 3115) parameter returns the generated data packet.

See Also

GetRecords (see page 3109)

DataSet (see page 3160)

Options (see page 3087)

58.1.7.4.3 TDataSetProvider.CreateResolver Method

Creates a resolver component for handling the application of updates.

Pascal

```
function CreateResolver: TCustomResolver; override;
```

C++

```
virtual __fastcall TCustomResolver CreateResolver();
```

Description

CreateResolver is called automatically when a resolver is needed to implement one of the provider's methods. CreateResolver returns a newly instantiated resolver object, which is assigned to the Resolver (see page 3088) property. The type of resolver created depends on the ResolveToDataSet (see page 3162) property:

When ResolveToDataSet (see page 3162) is true, CreateResolver returns a TDataSetResolver (see page 3176) component, which applies updates directly to the dataset specified by the DataSet (see page 3160) property.

When ResolveToDataSet (see page 3162) is false, CreateResolver returns a TSQLResolver (see page 3196) component, which applies updates directly to the database server associated with DataSet (see page 3160).

See Also

Resolver (see page 3088)

ResolveToDataSet (see page 3162)

58.1.7.4.4 TDataSetProvider.DoBeforeExecute Method

Generates a BeforeExecute (see page 3169) event.

Pascal

```
procedure DoBeforeExecute(const CommandText: WideString; var Params: OleVariant; var
OwnerData: OleVariant); override;
```

C++

```
virtual __fastcall DoBeforeExecute(const BSTR CommandText, OleVariant Params, OleVariant
OwnerData);
```

Description

The Execute (see page 3107) method calls DoBeforeExecute to generate the BeforeExecute (see page 3169) event before

it executes CommandText or executes an associated query or stored procedure.

CommandText is either an optional SQL statement that replaces the SQL of an associated query, or the name of a stored procedure that replaces the associated stored procedure. If Options (see page 3161) includes poAllowCommandText, DoBeforeExecute sends CommandText to the associated dataset so that it can execute the provided statement instead of its query or stored procedure.

Params (see page 3162) contains any parameter values that should be applied to CommandText or to the query or stored procedure before execution. DoBeforeExecute assigns these parameter values to the dataset (if appropriate).

OwnerData (see page 3162) supplies custom information to the BeforeExecute (see page 3169) event handler. The event handler can change this value. The returned value is then passed to an AfterExecute (see page 3165) event handler and then returned to the caller of the Execute (see page 3107) method.

See Also

Execute (see page 3107)

BeforeExecute (see page 3121)

AfterExecute (see page 3117)

DataSet (see page 3160)

58.1.7.4.5 TDataSetProvider.DoBeforeGetRecords Method

Generates a BeforeGetRecords (see page 3170) event.

Pascal

```
procedure DoBeforeGetRecords(Count: Integer; Options: Integer; const CommandText: WideString;
var Params: OleVariant; var OwnerData: OleVariant); override;
```

C++

```
virtual __fastcall DoBeforeGetRecords(int Count, int Options, const BSTR CommandText,
OleVariant Params, OleVariant OwnerData);
```

Description

The GetRecords (see page 3109) method calls DoBeforeGetRecords to generate the BeforeGetRecords (see page 3170) event before it fetches data and assembles it into a data packet.

Count is the number of records requested. TDataSetProvider (see page 3145) does not use this parameter.

Options (see page 3161) indicates whether the data packet should include metadata, whether it should start with the first record, and whether the data packet should be encoded in XML. In TDataSetProvider (see page 3145), DoBeforeGetRecords ignores this parameter.

CommandText specifies either an optional SQL statement that replaces the SQL of an associated query, or the name of a table or stored procedure that replaces the associated table or stored procedure. If Options (see page 3161) includes poAllowCommandText, DoBeforeGetRecords sends CommandText to the associated dataset so that it can generate the set of records.

Params (see page 3162) contains any parameter values that should be applied to CommandText or to a query or stored procedure before fetching data into a data packet. DoBeforeGetRecords assigns these values to the associated dataset, if appropriate.

OwnerData (see page 3162) supplies custom information to the BeforeGetRecords (see page 3170) event handler. The event handler can change this value. The returned value is then passed to an AfterGetRecords (see page 3166) event handler and then (optionally) returned to the caller of the GetRecords (see page 3109) method.

See Also

GetRecords (see page 3109)

BeforeGetRecords (see page 3122)

AfterGetRecords (see page 3118)

DataSet (see page 3160)

58.1.7.4.6 TDataSetProvider.DoGetProviderAttributes Method

Fills a list with information about the dataset that can be used when providing or resolving.

Pascal

```
procedure DoGetProviderAttributes(DataSet: TDataSet; List: TList); virtual;
```

C++

```
virtual __fastcall DoGetProviderAttributes(TDataSet DataSet, TList List);
```

Description

DoGetProviderAttributes is used internally to fetch information about the dataset. This includes both information provided by the dataset (using the IProviderSupport method PSGetAttributes) and custom information supplied by an OnGetDataSetProperties (see page 3174) event handler.

The DataSet (see page 3160) parameter specifies the dataset that stores the information. The List parameter is the list that is filled by DoGetProviderAttributes. DoGetProviderAttributes adds information to the list in Name/Value pairs.

See Also

PSGetAttributes

OnGetDataSetProperties (see page 3174)

58.1.7.4.7 TDataSetProvider.DoGetTableName Method

Generates an OnGetTableName (see page 3175) event.

Pascal

```
procedure DoGetTableName(DataSet: TDataSet; var TableName: WideString); virtual;
```

C++

```
virtual __fastcall DoGetTableName(TDataSet DataSet, BSTR TableName);
```

Description

The resolver calls DoGetTableName when it initializes the information it needs to apply updates. Override this method to provide an alternate mechanism for determining the name of the dataset that is the target of an update operation.

DataSet (see page 3160) is the dataset that is the target of the update operation.

The name of the target dataset as it should appear in generated SQL statements is returned as the TableName parameter.

See Also

OnGetTableName (see page 3175)

Resolver (see page 3088)

58.1.7.4.8 TDataSetProvider.FetchDetails Method

Fills a delta dataset with all nested dataset field values.

Pascal

```
procedure FetchDetails(Source: TDataSet; Delta: TDataSet); override;
```

C++

```
virtual __fastcall FetchDetails(TDataSet Source, TDataSet Delta);
```

Description

FetchDetails is used internally to fetch the current values of any nested datasets when the provider is executing its RowRequest (see page 3113) method. Source is a source dataset that has the current values for all fields. Delta is a delta dataset that must be updated so that its nested detail fields match the values in Source.

See Also

RowRequest (see page 3113)

58.1.7.4.9 TDataSetProvider.FindRecord Method

Pascal

```
function FindRecord(Source: TDataSet; Delta: TDataSet; UpdateMode: TUpdateMode): Boolean;
virtual;
```

C++

```
virtual __fastcall Boolean FindRecord(TDataSet Source, TDataSet Delta, TUpdateMode UpdateMode);
```

Description

This is FindRecord, a member of class TDataSetProvider.

58.1.7.4.10 TDataSetProvider.GetDataSetFromDelta Method

Returns a dataset positioned on the record of the source dataset that is currently being fetched or updated.

Pascal

```
function GetDataSetFromDelta(ATree: TUpdateTree; Source: TDataSet; Delta: TDataSet; Mode:
TUpdateMode): TDataSet; virtual;
```

C++

```
virtual __fastcall TDataSet GetDataSetFromDelta(TUpdateTree ATree, TDataSet Source, TDataSet
Delta, TUpdateMode Mode);
```

Description

GetDataSetFromDelta is used internally to fetch a specific record when the source dataset is unidirectional.

ATree is a TUpdateTree (see page 3205) object that represents the update or data fetch operation.

Source is a source dataset. Because it is unidirectional, however, it is not necessarily positioned on the correct record.

Delta is a delta dataset that indicates the desired record.

Mode indicates the criteria to use for locating a record on the server from the dataset that Source uses. The values that follow these criteria come from Delta.

GetDataSetFromDelta returns a unidirectional dataset that corresponds to Source but that contains a single record based on the

criteria in Delta and Mode.

See Also

IsUniDirectional

DataSet (see page 3160)

58.1.7.4.11 TDataSetProvider.InternalApplyUpdates Method

Provides the underlying implementation of the ApplyUpdates (see page 3100) method.

Pascal

```
function InternalApplyUpdates(const Delta: OleVariant; MaxErrors: Integer; out ErrorCount:
Integer): OleVariant; override;
```

C++

```
virtual __fastcall OleVariant InternalApplyUpdates(const OleVariant Delta, int MaxErrors, int
ErrorCount);
```

Description

The ApplyUpdates (see page 3100) method calls InternalApplyUpdates to apply the updates contained in a delta packet after generating the BeforeApplyUpdates (see page 3168) event and before generating the AfterApplyUpdates (see page 3164) event.

If the Options (see page 3161) property includes poReadOnly, InternalApplyUpdates raises an exception. Otherwise, it tells the dataset to start a transaction, calls the ApplyUpdates (see page 3100) method of its Resolver (see page 3088) component to handle the actual update and resolving process, and tells the dataset to complete the transaction.

Delta is the delta packet containing insertions, deletions, and modifications to be applied.

MaxErrors indicates the maximum number of errors that can be applied before stopping the update operation. If MaxErrors is –1, the provider tries to apply all updates.

ErrorCount returns the number of errors encountered during the update operation.

InternalApplyUpdates returns a data packet of updates that could not be applied to the database as an OleVariant.

See Also

Resolver (see page 3088)

ApplyUpdates (see page 3128)

CreateResolver (see page 3150)

Options (see page 3087)

ApplyUpdates (see page 3100)

58.1.7.4.12 TDataSetProvider.InternalExecute Method

Provides the underlying implementation of the Execute method.

Pascal

```
procedure InternalExecute(const CommandText: WideString; var Params: OleVariant); override;
```

C++

```
virtual __fastcall InternalExecute(const BSTR CommandText, OleVariant Params);
```

Description

The Execute method calls InternalExecute to execute the appropriate SQL statement after calling DoBeforeExecute to generate a BeforeExecute event and apply the CommandText and Params. As Implemented in TCustomProvider, InternalExecute does nothing. Descendants override this method to pass the execute command on to an associated dataset.

CommandText is an SQL statement that replaces the SQL of an associated query, or the name of a stored procedure that replaces the associated stored procedure. This parameter is ignored if the Options property does not include poAllowCommandText.

On entry, Params supplies parameter values for the query, stored procedure, or CommandText string. On exit, Params returns any output parameters.

See Also

Execute (see page 3107)

DoBeforeExecute (see page 3105)

58.1.7.4.13 TDataSetProvider.InternalGetParams Method

Returns the requested parameter values as a Variant array.

Pascal

```
function InternalGetParams(Types: TParamTypes = AllParamTypes): OleVariant; override;
```

C++

```
virtual __fastcall OleVariant InternalGetParams(TParamTypes Types = AllParamTypes);
```

Description

The provider uses InternalGetParams internally whenever it needs to assemble the current parameter values into an OleVariant. This provides the underlying implementation of the GetParams (see page 3108) method, as well as assembling parameter values for methods that return output parameters (such as Execute (see page 3107) or GetRecords (see page 3109)).

The Types parameter limits the parameter values that are returned to a subset of all possible parameter types. If this parameter is omitted, all parameter values are returned.

See Also

GetParams (see page 3108)

Execute (see page 3107)

GetRecords (see page 3109)

DataSet (see page 3160)

58.1.7.4.14 TDataSetProvider.InternalGetRecords Method

Provides the underlying implementation of the GetRecords (see page 3109) method.

Pascal

```
function InternalGetRecords(Count: Integer; out RecsOut: Integer; Options: TGetRecordOptions;
const CommandText: WideString; var Params: OleVariant): OleVariant; override;
```

C++

```
virtual __fastcall OleVariant InternalGetRecords(int Count, int RecsOut, TGetRecordOptions
Options, const BSTR CommandText, OleVariant Params);
```

Description

The GetRecords (see page 3109) method calls InternalGetRecords to assemble a data packet after calling DoBeforeGetRecords (see page 3151) to apply the CommandText and Params (see page 3162) and generate a BeforeGetRecords (see page 3170) event. InternalGetRecords calls CreateDataPacket (see page 3149) to create a data packet that reflects the current property settings of the provider. It then generates an OnGetData (see page 3174) event.

Count indicates the number of records to retrieve. If Count is -1, all records are retrieved. If Count is 0, only metadata is retrieved. If Count is greater than 0, only the specified number of records are retrieved.

RecsOut returns the actual number of records retrieved.

Options (see page 3161) indicates what information should be added to the data packet in addition to data. It can include grMetaData (include metadata), grReset (start providing records with the first one in the dataset), and grXML (encode the datapacket in XML rather than as an OleVariant) or grXMLUTF8 (same as grXML but use UTF8 to encode extended characters).

CommandText specifies an optional SQL statement that replaces the SQL of an associated query, or the name of a table or stored procedure that replaces the associated table or stored procedure. This parameter is ignored if the Options (see page 3161) property does not include poAllowCommandText.

Params (see page 3162) specifies any parameter values that should be applied to CommandText or to an associated query or stored procedure. If the provider is associated with a TTable, Params (see page 3162) specifies field values that filter the result set. On exit from GetRecords (see page 3109), Params (see page 3162) returns any output parameters.

Records are returned as a data packet in a Variant.

Note: If the DataSet (see page 3160) property is not set, InternalGetRecords raises an exception.

See Also

Data (see page 3115)

CreateDataPacket (see page 3149)

OnGetData (see page 3091)

BeforeGetRecords (see page 3122)

AfterGetRecords (see page 3118)

DoBeforeGetRecords (see page 3151)

58.1.7.4.15 TDataSetProvider.InternalRowRequest Method

Provides the underlying implementation of the RowRequest (see page 3113) method.

Pascal

```
function InternalRowRequest(const Row: OleVariant; Options: TFetchOptions): OleVariant;
override;
```

C++

```
virtual __fastcall OleVariant InternalRowRequest(const OleVariant Row, TFetchOptions Options);
```

Description

The RowRequest (see page 3113) method calls InternalRowRequest to assemble the specified information after generating a BeforeRowRequest (see page 3172) event and before generating an AfterRowRequest (see page 3167) event. InternalRowRequest passes this request on to its associated Resolver (see page 3088).

The Row parameter is an OleVariant that describes the record about which information is requested.

InternalRowRequest returns the requested information in a delta packet.

See Also

RowRequest (see page 3113)

Resolver (see page 3088)

RowRequest (see page 3134)

58.1.7.4.16 TDataSetProvider.LocateRecord Method

Positions a data set on the record that corresponds to the current update in a specified delta packet.

Pascal

```
procedure LocateRecord(Source: TDataSet; Delta: TDataSet); override;
```

C++

```
virtual __fastcall LocateRecord(TDataSet Source, TDataSet Delta);
```

Description

LocateRecord is called indirectly by the RowRequest (see page 3113) method to locate a record before fetching information about it. The Source parameter is the dataset containing current record values. The Delta parameter is a delta packet that identifies the desired type of information and information identifying the desired record.

LocateRecord uses the criteria specified by UpdateMode (see page 3163) to identify the desired record.

See Also

RowRequest (see page 3113)

UpdateMode (see page 3089)

58.1.7.4.17 TDataSetProvider.Notification Method

Responds to notifications that objects are being created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification overrides the inherited method to add a check that the dataset specified by the DataSet (see page 3160) property is not being deleted. If DataSet (see page 3160) is about to be destroyed, Notification sets the DataSet (see page 3160) property to nil (Delphi) or NULL (C++).

See Also

OnUpdateError (see page 3092)

Create

58.1.7.4.18 TDataSetProvider.Reset Method

Pascal

```
procedure Reset;
```

C++

```
__fastcall Reset();
```

Description

This is Reset, a member of class TDataSetProvider.

58.1.7.4.19 TDataSetProvider.SetCommandText Method

Pascal

```
procedure SetCommandText(const CommandText: WideString); virtual;
```

C++

```
virtual __fastcall SetCommandText(const BSTR CommandText);
```

Description

This is SetCommandText, a member of class TDataSetProvider.

58.1.7.4.20 TDataSetProvider.SetParams Method

Pascal

```
procedure SetParams(Values: OleVariant); virtual;
```

C++

```
virtual __fastcall SetParams(OleVariant Values);
```

Description

This is SetParams, a member of class TDataSetProvider.

58.1.7.4.21 TDataSetProvider.UpdateRecord Method

Retrieves specified data from a source dataset into a delta packet.

Pascal

```
procedure UpdateRecord(Source: TDataSet; Delta: TDataSet; BlobsOnly: Boolean; KeyOnly:
Boolean); override;
```

C++

```
virtual __fastcall UpdateRecord(TDataSet Source, TDataSet Delta, Boolean BlobsOnly, Boolean
KeyOnly);
```

Description

UpdateRecord is called indirectly by the RowRequest (see page 3113) method to fetch information about a specified record. The Source parameter is the dataset containing indicated record. The Delta parameter is a delta packet that returns the requested data. BlobsOnly indicates whether all field values should be provided (BlobsOnly == false) or only Blob field values (BlobsOnly == true).

See Also

RowRequest (see page 3113)

58.1.7.5 TDataSetProvider Properties

The properties of the TDataSetProvider class are listed here.

Protected Properties

	Name	Description
	TransactionStarted (see page 3163)	This is TransactionStarted, a member of class TDataSetProvider.

Public Properties

	Name	Description
	ActiveUpdateException (see page 3159)	This is ActiveUpdateException, a member of class TDataSetProvider.
	OwnerData (see page 3162)	This is OwnerData, a member of class TDataSetProvider.
R	Params (see page 3162)	Stores the parameter values currently in use.

Published Properties

	Name	Description
	Constraints (see page 3159)	Controls whether database constraints are passed to the client so that it can check them locally.
	DataSet (see page 3160)	Specifies the dataset from which the provider provides data and to which it applies updates.
	Exported (see page 3160)	Allows client applications to pass calls to the provider when it resides on an application server.
	Options (see page 3161)	Influences what is included in data packets and how the information in data packets is used.
	ResolveToDataSet (see page 3162)	Specifies whether updates should be applied to a dataset or directly to the database server.
	UpdateMode (see page 3163)	Controls how the provider locates records when applying updates.

58.1.7.5.1 TDataSetProvider.ActiveUpdateException Property

Pascal

```
property ActiveUpdateException: Exception;
```

C++

```
__property Exception ActiveUpdateException;
```

Description

This is ActiveUpdateException, a member of class TDataSetProvider.

58.1.7.5.2 TDataSetProvider.Constraints Property

Controls whether database constraints are passed to the client so that it can check them locally.

Pascal

```
property Constraints: Boolean;
```

C++

```
__property Boolean Constraints;
```

Description

Set Constraints to specify whether data constraints defined for the provider's associated dataset are sent along with the metadata for the dataset. Sending constraints allows client datasets to check for constraint violations locally, so that updates are less likely to fail when sent back to the database server.

When Constraints is true, constraints are sent. To prevent the provider from sending constraint information, set Constraints to false.

Note: Before constraints can be sent to client applications, they must be imported to the data dictionary on the application server. Use SQL explorer to import constraints to the data dictionary.

See Also

Constraints

DataSet (see page 3160)

58.1.7.5.3 TDataSetProvider.DataSet Property

Specifies the dataset from which the provider provides data and to which it applies updates.

Pascal

```
property DataSet: TDataSet;
```

C++

```
__property TDataSet DataSet;
```

Description

Set DataSet to link the dataset provider to a dataset that it represents. The dataset provider packages the records from (and other information about) DataSet into data packets that can be used by client datasets and XML brokers. DataSet must be defined if the provider is used to create data packets.

If the ResolveToDataSet (see page 3162) property is true, the provider applies updates from the application server directly to this dataset. Otherwise, the provider applies updates directly to the underlying database server.

Note: The provider communicates with DataSet using the IProviderSupport interface. This interface is available in all datasets, although in TDataSet most of the IProviderSupport methods are simple stubs that either do nothing or generate an exception. The dataset classes that ship with the IDE (dbExpress datasets, BDE-enabled datasets, ADO-enabled datasets, InterBase Express datasets,

and client datasets) override these protected methods to implement the IProviderSupport interface in a more useful fashion. If you are using custom dataset components, check with the vendor whether they can work with a dataset provider.

See Also

ApplyUpdates (see page 3100)

Data (see page 3115)

IProviderSupport

Connecting a Client Dataset to Another Dataset in the Same Application

Determining the Source of Data

58.1.7.5.4 TDataSetProvider.Exported Property

Allows client applications to pass calls to the provider when it resides on an application server.

Pascal

```
property Exported: Boolean;
```

C++

```
__property Boolean Exported;
```

Description

Use Exported to indicate whether the provider is made available to clients of a remote data module that owns it. When Exported is true, client applications can specify the provider as a target for calls made using the IAppServer interface. When Exported is false, IAppServer calls that specify the provider cause an exception.

Note: In addition to the Exported property, the provider must be registered with the remote data module for clients to access it using the data module's IAppServer interface. When the provider's Owner is the remote data module, it is automatically registered. If the provider has a different owner, the server application must explicitly register providers using the remote data module's RegisterProvider method.

See Also

IAppServer

Providers

RegisterProvider

58.1.7.5.5 TDataSetProvider.Options Property

Influences what is included in data packets and how the information in data packets is used.

Pascal

```
property Options: TProviderOptions;
```

C++

```
__property TProviderOptions Options;
```

Description

Set Options to customize how the provider communicates with client datasets. Options is a set that specifies

Whether nested detail sets and BLOB data are included in data packets or fetched separately.

Whether data packets include field component properties such as formats, display names, and maximum & minimum values.

Whether data packets are provided on a read-only basis, and whether specific types of changes (insertions, edits, or deletions) are disabled.

Whether changes to master table fields should cause corresponding updates to detail tables.

Whether a single update can affect multiple server records.

Whether the client's records are refreshed when it applies updates.

Whether the client can send SQL statements that override the provider's dataset.

Note: Not all options are supported by an InternetExpress client. Specifically, InternetExpress clients don't support poFetchBlobsOnDemand, poFetchDetailsOnDemand, poDisableInserts, poDisableEdits, psDisableDeletes, poPropogateChanges, and poAllowCommandText.

See Also

Data (see page 3115)

ApplyUpdates (see page 3100)

Overriding the Dataset On the Application Server

Setting Options That Influence the Data Packets

58.1.7.5.6 TDataSetProvider.OwnerData Property

Pascal

```
property OwnerData: OleVariant;
```

C++

```
__property OleVariant OwnerData;
```

Description

This is OwnerData, a member of class TDataSetProvider.

58.1.7.5.7 TDataSetProvider.Params Property

Stores the parameter values currently in use.

Pascal

```
property Params: TParams;
```

C++

```
__property TParams Params;
```

Description

This protected property acts as a temporary cache for parameter values from clients. It is set to the parameter values supplied by calls to the Execute (see page 3107) or GetRecords (see page 3109) method.

Note: If the dataset provider is responding to multiple client datasets or XML brokers

, the value of Params can only be used within the context of a single call to SetParams (see page 3158), Execute (see page 3107), or GetRecords (see page 3109). After that, other client calls can change the value of parameters.

See Also

Execute (see page 3107)

GetRecords (see page 3109)

58.1.7.5.8 TDataSetProvider.ResolveToDataSet Property

Specifies whether updates should be applied to a dataset or directly to the database server.

Pascal

```
property ResolveToDataSet: Boolean;
```

C++

```
__property Boolean ResolveToDataSet;
```

Description

Set ResolveToDataSet to specify how updates are applied. When ResolveToDataSet is true, the Resolver (see page 3088) property is set to a TDataSetResolver (see page 3176) component, which applies updates directly to the dataset specified by the DataSet (see page 3160) property. This can be useful if the application uses the events on the dataset component or if the dataset does not represent the data from a database server (for example, a client dataset).

When ResolveToDataSet is false, the Resolver (see page 3088) property is set to a TSQLResolver (see page 3196) component, which applies updates directly to the database server associated with DataSet (see page 3160). This can be more efficient, because it skips the intermediate step of using the dataset. It is the only way to resolve updates if the source dataset is read-only (for example, a unidirectional dataset).

See Also

DataSet (see page 3160)

Resolver (see page 3088)

Choosing How to Apply Updates Using a Datasetm Provider

58.1.7.5.9 TDataSetProvider.TransactionStarted Property

Pascal

```
property TransactionStarted: Boolean;
```

C++

```
__property Boolean TransactionStarted;
```

Description

This is TransactionStarted, a member of class TDataSetProvider.

58.1.7.5.10 TDataSetProvider.UpdateMode Property

Controls how the provider locates records when applying updates.

Pascal

```
property UpdateMode: TUpdateMode;
```

C++

```
__property TUpdateMode UpdateMode;
```

Description

Use UpdateMode to specify the criteria for locating a record in the dataset when applying updates. UpdateMode specifies whether modified records are located based on all columns (fields), on only the key fields, or on the key fields plus the original values of fields that have been modified.

Note: To control how the provider locates records by specifying individual fields, use the field components' ProviderFlags property.

See Also

ApplyUpdates (see page 3100)

ProviderFlags

Influencing How Updates Are Applied

58.1.7.6 TDataSetProvider Events

The events of the TDataSetProvider class are listed here.

Published Events

	Name	Description
	AfterApplyUpdates (see page 3164)	Occurs after the provider applies updates.
	AfterExecute (see page 3165)	Occurs after the provider passes an execute command to its dataset.
	AfterGetParams (see page 3166)	Occurs after the provider creates an OleVariant containing parameter values to send to the client dataset.
	AfterGetRecords (see page 3166)	Occurs after the provider creates a data packet to send to the client dataset.
	AfterRowRequest (see page 3167)	Occurs after the provider fetches new information about the current record.
	AfterUpdateRecord (see page 3168)	Occurs when the associated resolver has just updated a record.
	BeforeApplyUpdates (see page 3168)	Occurs before the provider applies updates from a client dataset.
	BeforeExecute (see page 3169)	Occurs before the provider passes an execute command to its dataset.
	BeforeGetParams (see page 3170)	Occurs before the provider creates an OleVariant that encodes the current parameter values and sends them to a client dataset.
	BeforeGetRecords (see page 3170)	Occurs before the provider creates a data packet and sends it to a client dataset.
	BeforeRowRequest (see page 3172)	Occurs before the provider creates a delta packet with information about the current record.
	BeforeUpdateRecord (see page 3172)	Occurs immediately before each record is applied to the remote dataset.
	OnDataRequest (see page 3173)	Occurs when a provider calls its DataRequest method.
	OnGetData (see page 3174)	Occurs after a provider retrieves data but before it is sent to the client.
	OnGetDataSetProperties (see page 3174)	Occurs when the provider adds additional information to a data packet it is creating.
	OnGetTableName (see page 3175)	Occurs when a resolver initializes its information about the table to which it applies updates.
	OnUpdateData (see page 3175)	Occurs when a provider begins to apply updates.
	OnUpdateError (see page 3176)	Occurs when a provider can't update a record because of an error condition.
	OnValidate (see page 3176)	This is OnValidate, a member of class TDataSetProvider.

58.1.7.6.1 TDataSetProvider.AfterApplyUpdates Event

Occurs after the provider applies updates.

Pascal

```
property AfterApplyUpdates: TRemoteEvent;
```

C++

```
__property TRemoteEvent AfterApplyUpdates;
```

Description

Write an AfterApplyUpdates event handler to return custom information to a client dataset after applying updates. AfterApplyUpdates is part of the mechanism by which a client dataset and a provider communicate information about updates. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset calls the AS_ApplyUpdates method of the IAppServer interface, the following events occur:

1.The provider receives a BeforeApplyUpdates event, where it can respond to or change custom information that the client dataset sends.

2.If it is a dataset provider, the provider receives an OnUpdateData event where it can edit the updates before they are applied.

3. The provider applies the updates, and attempts to resolve problems.

4.The provider receives an AfterApplyUpdates event, where it can encode custom information into its OwnerData parameter or respond to information from the BeforeApplyUpdates event handler.

5.The client dataset receives an AfterApplyUpdates event, where it can respond to the custom information returned by the

provider in its AfterApplyUpdates event handler.

Note: If ApplyUpdates is called without the OwnerData parameter, on entry to the event handler the OwnerData parameter contains the value set in the BeforeApplyUpdates event handler and it is ignored on exit from the AfterApplyUpdates event handler.

See Also

ApplyUpdates (see page 3100)

AS_ApplyUpdates

ApplyUpdates

AfterApplyUpdates

BeforeApplyUpdates (see page 3120)

OnUpdateData (see page 3092)

Responding to Client Update Requests

Communicating with the Client Dataset

58.1.7.6.2 TDataSetProvider.AfterExecute Event

Occurs after the provider passes an execute command to its dataset.

Pascal

```
property AfterExecute: TRemoteEvent;
```

C++

```
__property TRemoteEvent AfterExecute;
```

Description

Write an AfterExecute event handler to return custom information to a client dataset after executing a query or stored procedure. AfterExecute is part of the mechanism by which a client dataset and a provider communicate information when executing queries or stored procedures that do not return a cursor. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset calls the AS_Execute method of the IAppServer interface, the following events occur:

1.The provider receives a BeforeExecute event, where it can respond to or change custom information that the client dataset sends.

2.The provider dispatches its Execute method to execute a query or stored procedure.

3.The provider receives an AfterExecute event, where it can encode custom information into its OwnerData parameter or respond to information from the BeforeExecute event handler.

4.The client dataset receives an AfterExecute event, where it can respond to the custom information returned by the provider in its AfterExecute event handler.

See Also

Execute (see page 3107)

AS_Execute

Execute

BeforeExecute (see page 3121)

DataSet (see page 3160)

AfterExecute

Communicating with the Client Dataset

58.1.7.6.3 TDataSetProvider.AfterGetParams Event

Occurs after the provider creates an OleVariant containing parameter values to send to the client dataset.

Pascal

```
property AfterGetParams: TRemoteEvent;
```

C++

```
__property TRemoteEvent AfterGetParams;
```

Description

Write an AfterGetParams event handler to send custom information to a client dataset when returning parameters from the provider's dataset. AfterGetParams is part of the mechanism by which a provider and a client dataset communicate information about parameter values. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset calls the AS_GetParams method of the IAppServer interface, the following events occur:

1.The provider receives a BeforeGetParams event, where it can respond to or change custom information that the client dataset sends.

2.The provider looks up the parameter values, if appropriate, and encodes them in an OleVariant.

3.The provider receives an AfterGetParams event, where it can encode custom information into its OwnerData parameter or respond to information from the BeforeGetParams event handler.

4.The client dataset receives an AfterGetParams event, where it can respond to the custom information returned by the provider in its AfterGetParams event handler.

See Also

AS_GetParams

GetParams (see page 3108)

AfterGetParams

BeforeGetParams (see page 3121)

58.1.7.6.4 TDataSetProvider.AfterGetRecords Event

Occurs after the provider creates a data packet to send to the client dataset.

Pascal

```
property AfterGetRecords: TRemoteEvent;
```

C++

```
__property TRemoteEvent AfterGetRecords;
```

Description

Write an AfterGetRecords event handler to send custom information to a client dataset when providing a data packet. AfterGetRecords is part of the mechanism by which a provider and a client dataset communicate information about data fetching. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to

communicate persistent state information.

When the client dataset calls the AS_GetRecords method of the IAppServer interface, the following events occur:

1.The provider receives a BeforeGetRecords event, where it can respond to or change custom information that the client dataset sends.

2.If it is a dataset provider, the provider receives an OnGetData event where it can edit the data before creating a data packet, and then generates the data packet.

3.The provider receives an AfterGetRecords event, where it can encode custom information into its OwnerData parameter or respond to information from the BeforeGetRecords event handler.

4.The client dataset receives an AfterGetRecords event, where it can respond to the custom information returned by the provider in its AfterGetRecords event handler.

See Also

AS_GetRecords

OnGetData (see page 3091)

GetRecords (see page 3109)

AfterGetRecords

BeforeGetRecords (see page 3122)

Communicating with the Client Dataset

Responding to Client Data Requests

58.1.7.6.5 TDataSetProvider.AfterRowRequest Event

Occurs after the provider fetches new information about the current record.

Pascal

```
property AfterRowRequest: TRemoteEvent;
```

C++

```
__property TRemoteEvent AfterRowRequest;
```

Description

Write an AfterRowRequest event handler to send custom information to a client dataset along with information about the current record. AfterRowRequest is part of the mechanism by which a provider and a client dataset communicate information about data fetching. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset calls the AS_RowRequest method of the IAppServer interface, the following events occur:

1.The provider receives a BeforeRowRequest event, where it can respond to or change custom information that the client dataset sends.

2.The provider creates a delta packet that contains the requested information.

3.The provider receives an AfterRowRequest event, where it can encode custom information into its OwnerData parameter or respond to information from the BeforeRowRequest event handler.

4.The client dataset receives an AfterRowRequest event, where it can respond to the custom information returned by the provider in its AfterRowRequest event handler.

See Also

AS_RowRequest

RowRequest (see page 3113)

AfterRowRequest

BeforeRowRequest (see page 3123)

FetchBlobs

FetchParams

RefreshRecord

Communicating with the Client Dataset

58.1.7.6.6 TDataSetProvider.AfterUpdateRecord Event

Occurs when the associated resolver has just updated a record.

Pascal

```
property AfterUpdateRecord: TAfterUpdateRecordEvent;
```

C++

```
__property TAfterUpdateRecordEvent AfterUpdateRecord;
```

Description

Write an AfterUpdateRecord event handler to provide special processing immediately after a record has been successfully updated. For example, use AfterUpdateRecord to maintain a log of successful updates.

See Also

BeforeUpdateRecord (see page 3090)

AfterApplyUpdates (see page 3116)

58.1.7.6.7 TDataSetProvider.BeforeApplyUpdates Event

Occurs before the provider applies updates from a client dataset.

Pascal

```
property BeforeApplyUpdates: TRemoteEvent;
```

C++

```
__property TRemoteEvent BeforeApplyUpdates;
```

Description

Write a BeforeApplyUpdates event handler to respond to custom information from a client dataset. BeforeApplyUpdates is part of the mechanism by which a provider and a client dataset communicate information about updates. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset's ApplyUpdates method is called, the following events occur:

1.The client dataset receives a BeforeApplyUpdates event, where it can encode custom information into an OleVariant.

2.The provider receives a BeforeApplyUpdates event, where the OleVariant from the client dataset appears as the OwnerData parameter. The provider can respond to or change that information before applying the updates.

3.If it is a dataset provider, the provider applies the updates, and receives an OnUpdateError event for any problem records.

4.The provider receives an AfterApplyUpdates event, where it can encode custom information into its OwnerData parameter or respond to information from the BeforeApplyUpdates event handler.

5.The client dataset receives an AfterApplyUpdates event, where it can respond to the custom information returned by the provider in its AfterApplyUpdates event handler.

Note: If ApplyUpdates is called without the OwnerData parameter, the OwnerData parameter is nil (Delphi) or NULL (C++) on entry to the event handler and its value (on exit) is only passed to the AfterApplyUpdates event handler but not returned to the caller.

See Also

ApplyUpdates (see page 3100)

AS_ApplyUpdates

ApplyUpdates

BeforeApplyUpdates

AfterApplyUpdates (see page 3116)

Responding to Client Update Requests

Communicating with the Client Dataset

58.1.7.6.8 TDataSetProvider.BeforeExecute Event

Occurs before the provider passes an execute command to its dataset.

Pascal

```
property BeforeExecute: TRemoteEvent;
```

C++

```
__property TRemoteEvent BeforeExecute;
```

Description

Write a BeforeExecute event handler to respond to custom information from a client dataset before executing a query or stored procedure. Any parameter values supplied by the client dataset with the Execute command are applied before this event.

BeforeExecute is part of the mechanism by which a provider and a client dataset communicate information when executing queries or stored procedures that do not return cursors. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset's Execute method is called, the following events occur:

1.The client dataset receives a BeforeExecute event, where it can encode custom information into an OleVariant.

2.The provider receives a BeforeExecute event, where the OleVariant from the client dataset appears as the OwnerData parameter. The provider can respond to or change that information before passing the execute command on to its dataset.

3.The provider dispatches its Execute method to execute a query or stored procedure.

4.The provider receives an AfterExecute event, where it can encode custom information into its OwnerData parameter or respond to information from the BeforeExecute event handler.

5.The client dataset receives an AfterExecute event, where it can respond to the custom information returned by the provider in its AfterExecute event handler.

See Also

Execute (see page 3107)

AS_Execute

Execute

DataSet (see page 3160)

BeforeExecute

58.1.7.6.9 TDataSetProvider.BeforeGetParams Event

Occurs before the provider creates an OleVariant that encodes the current parameter values and sends them to a client dataset.

Pascal

```
property BeforeGetParams: TRemoteEvent;
```

C++

```
__property TRemoteEvent BeforeGetParams;
```

Description

Write a BeforeGetParams event handler to respond to custom information from a client dataset. BeforeGetParams is part of the mechanism by which a provider and a client dataset communicate information during parameter fetching. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset fetches parameter values from the provider, the following events occur:

1.The client dataset receives a BeforeGetParams event, where it can encode custom information into an OleVariant.

2.The provider receives a BeforeGetParams event, where the OleVariant from the client dataset appears as the OwnerData parameter. The provider can respond to or change that information before it fetches and encodes the parameter values.

3.The provider generates an OleVariant for the current parameter values of its dataset, if appropriate.

4.The provider receives an AfterGetParams event, where it can encode custom information into its OwnerData parameter or respond to information from the BeforeGetParams event handler.

5.The client dataset receives an AfterGetParams event, where it can respond to the custom information returned by the provider in its AfterGetParams event handler.

See Also

AS_GetParams

GetParams (see page 3108)

BeforeGetParams

AfterGetParams (see page 3118)

Communicating with the Client Dataset

58.1.7.6.10 TDataSetProvider.BeforeGetRecords Event

Occurs before the provider creates a data packet and sends it to a client dataset.

Pascal

```
property BeforeGetRecords: TRemoteEvent;
```

C++

```
__property TRemoteEvent BeforeGetRecords;
```

Description

Write a BeforeGetRecords event handler to respond to custom information from a client dataset. BeforeGetRecords is part of the mechanism by which a provider and a client dataset communicate information about data fetching. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset fetches data from the provider, the following events occur:

1.The client dataset receives a BeforeGetRecords event, where it can encode custom information into an OleVariant.

2.The provider receives a BeforeGetRecords event, where the OleVariant from the client dataset appears as the OwnerData parameter. The provider can respond to or change that information before it creates a data packet.

3.The provider generates the data packet.

4.The provider receives an AfterGetRecords event, where it can encode custom information into its OwnerData parameter or respond to information from the BeforeGetRecords event handler.

5.The client dataset receives an AfterGetRecords event, where it can respond to the custom information returned by the provider in its AfterGetRecords event handler.

Delphi Examples:

```
{
This example shows the implementation of a provider BeforeGetRecords
event handler and a RemoteDataModule UpdateRegistry method.  The
complete project using this code example is in
Demos/DelphiWin32/VCLWin32/MIDAS/Pooler.
}
procedure TPooledRDM.DataSetProvider1BeforeGetRecords(Sender: TObject;
  var OwnerData: OleVariant);
begin
  try
    Query1.Close;
    Query1.SQL.Text := OwnerData[0];
    if not VarIsNull(OwnerData[1]) and not VarIsEmpty(OwnerData[1]) then
    begin
      Query1.Open;
      if not Query1.Locate(Query1.Fields[0].FieldName, OwnerData[1], []) then
        raise Exception.Create('Record not found');
      Query1.Next;
    end;
  finally
    OwnerData := NULL;
  end;
end;
```

See Also

AS_GetRecords

GetRecords (see page 3109)

BeforeGetRecords

AfterGetRecords (see page 3118)

Communicating with the Client Dataset

Responding to Client Data Requests

58.1.7.6.11 TDataSetProvider.BeforeRowRequest Event

Occurs before the provider creates a delta packet with information about the current record.

Pascal

```
property BeforeRowRequest: TRemoteEvent;
```

C++

```
__property TRemoteEvent BeforeRowRequest;
```

Description

Write a BeforeRowRequest event handler to respond to custom information from a client dataset. BeforeRowRequest is part of the mechanism by which a provider and a client dataset communicate information about data fetching. When the provider is part of a stateless application server, this mechanism allows the provider and the client dataset to communicate persistent state information.

When the client dataset fetches detail information about the current record or refreshes the current record, the following events occur:

1.The client dataset receives a BeforeRowRequest event, where it can encode custom information into an OleVariant.

2.The provider receives a BeforeRowRequest event, where the custom information appears as the OwnerData parameter. The provider can respond to or change that information before it looks up the requested information.

3.The provider creates a delta packet that contains the requested information.

4.The provider receives an AfterRowRequest event, where it can encode custom information into its OwnerData parameter or respond to information from the BeforeRowRequest event handler.

5.The client dataset receives an AfterRowRequest event, where it can respond to the custom information returned by the provider in its AfterRowRequest event handler.

See Also

AS_RowRequest

RowRequest (see page 3113)

AfterRowRequest (see page 3119)

BeforeRowRequest

FetchBlobs

FetchParams

RefreshRecord

Communicating with the Client Dataset

58.1.7.6.12 TDataSetProvider.BeforeUpdateRecord Event

Occurs immediately before each record is applied to the remote dataset.

Pascal

```
property BeforeUpdateRecord: TBeforeUpdateRecordEvent;
```

C++

```
__property TBeforeUpdateRecordEvent BeforeUpdateRecord;
```

Description

Write a BeforeUpdateRecord event handler to:

Prepare individual record updates before they are applied to the source dataset or the database server. For example, you can insert additional changes into a record before it is updated or set the ProviderFlags property of the fields in the DeltaDS parameter to indicate how the provider should identify the records that are to be updated or changed and which fields should be updated.

Manually apply updates that the provider can't handle. For example, the provider can't generate the appropriate SQL when the data results from a multi-table join and more than one table must be updated. Write code in the event handler to apply the update and then set the Applied parameter to true so that the provider does not generate an error.

Selectively prevent certain updates from occurring. By setting the Applied parameter to true, the default update does not occur. When you prevent updates in this way, the provider does not generate an update error.

See Also

AfterUpdateRecord (see page 3090)

ApplyUpdates (see page 3100)

OnUpdateData (see page 3092)

OnUpdateError (see page 3092)

ProviderFlags

BeforeApplyUpdates (see page 3120)

58.1.7.6.13 TDataSetProvider.OnDataRequest Event

Occurs when a provider calls its DataRequest method.

Pascal

```
property OnDataRequest: TDataRequestEvent;
```

C++

```
__property TDataRequestEvent OnDataRequest;
```

Description

Write an OnDataRequest event handler to do custom processing of data. OnDataRequest has no built-in use. Instead, it allows developers to augment the IAppServer interface so that clients in a multi-tiered application can communicate directly with an individual provider.

If an application does not provide an OnDataRequest event handler, calls to DataRequest return NULL.

Note: To provide the same sort of custom processing on the client side, use an automation method on the connection component.

See Also

DataRequest (see page 3101)

IAppServer

Communicating with Providers Using Custom Events

Communicating with the Client Dataset

Responding to Client-generated Events

58.1.7.6.14 TDataSetProvider.OnGetData Event

Occurs after a provider retrieves data but before it is sent to the client.

Pascal

```
property OnGetData: TProviderDataEvent;
```

C++

```
__property TProviderDataEvent OnGetData;
```

Description

Write an OnGetData event handler to perform any custom data processing the provider should handle before returning data to a client. The OnGetData event occurs after a provider fetches data from its dataset but before the data is sent on to the client. Using the methods of the DataSet parameter, data can be edited in the OnGetData event handler before sending it to the client dataset or XML broker.

Note: OnGetData occurs when the Data property is read, as well as when the GetRecords method is called.

See Also

GetRecords (see page 3109)

OnUpdateData (see page 3092)

BeforeGetRecords (see page 3122)

AfterGetRecords (see page 3118)

Responding to Client Data Requests

58.1.7.6.15 TDataSetProvider.OnGetDataSetProperties Event

Occurs when the provider adds additional information to a data packet it is creating.

Pascal

```
property OnGetDataSetProperties: TGetDSProps;
```

C++

```
__property TGetDSProps OnGetDataSetProperties;
```

Description

Write an OnGetDataSetProperties event handler to include application-defined information in the data packets. This information can later be returned using the GetOptionalParam method of the client dataset that represents the data in the packet.

Note: The following names are reserved and can't be used as the name for storing custom information:

UNIQUE_KEY	DEFAULT_ORDER	CHANGE_LOG	SERVER_COL
CONSTRAINTS	DATASET_CONTEXT	DATASET_DELTA	LCID
BDERECORD_X	TABLE_NAME	MD_FIELDLINKS	UPDATEMODE

See Also

GetOptionalParam

SetOptionalParam

Adding Custom Information to Data Packets

58.1.7.6.16 TDataSetProvider.OnGetTableName Event

Occurs when a resolver initializes its information about the table to which it applies updates.

Pascal

```
property OnGetTableName: TGetTableNameEvent;
```

C++

```
__property TGetTableNameEvent OnGetTableName;
```

Description

Write an OnGetTableName event handler to enable the provider to apply updates to datasets where the resolver can't determine the target table from the provider's dataset. When the resolver component is initialized, it stores the name of the table that is the target of its updates. This table name can be used in generated SQL statements that apply the updates. When the dataset is a TTable, the resolver can use the table's TableName property. However, i If the dataset represents a multi-table query or stored procedure, the resolver has no way to determine the destination of the update without input from an OnGetTableName event handler.

Return the name of the target table as the TableName parameter. The resolver uses this to apply updates.

See Also

Resolver (see page 3088)

DataSet (see page 3160)

ApplyUpdates (see page 3100)

58.1.7.6.17 TDataSetProvider.OnUpdateData Event

Occurs when a provider begins to apply updates.

Pascal

```
property OnUpdateData: TProviderDataEvent;
```

C++

```
__property TProviderDataEvent OnUpdateData;
```

Description

Write an OnUpdateData event to handle special processing before the provider begins applying updates received from the client dataset or XML broker. Using the methods belonging to the DataSet parameter, an application can

Examine data (for example, for values or data changes that should not permitted), and raise exceptions that cancel applying of updates before they occur.

Change data (for example encrypting or decrypting values) before it is sent on to the source dataset or database server.

See Also

ApplyUpdates (see page 3100)

BeforeUpdateRecord (see page 3090)

Adding Custom Information to Data Packets

Editing Delta Packets Before Updating the Database

Influencing How Updates Are Applied

58.1.7.6.18 TDataSetProvider.OnUpdateError Event

Occurs when a provider can't update a record because of an error condition.

Pascal

```
property OnUpdateError: TResolverErrorEvent;
```

C++

```
__property TResolverErrorEvent OnUpdateError;
```

Description

Write an OnUpdateError event handler to respond to error conditions that arise when the provider applies a client dataset's updates. The OnUpdateError event handler is called for each record that can't be updated.

Warning: The provider can't detect when an update conflicts with another user's changes to a memo field. This situation will not generate an OnUpdateError event.

See Also

ApplyUpdates (see page 3100)

OnReconcileError

NewValue

OldValue

CurValue

Resolving Update Errors On the Provider

58.1.7.6.19 TDataSetProvider.OnValidate Event

Pascal

```
property OnValidate: TOnValidate;
```

C++

```
__property TOnValidate OnValidate;
```

Description

This is OnValidate, a member of class TDataSetProvider.

58.1.8 TDataSetResolver Class

TDataSetResolver applies updates to a dataset on behalf of a provider.

Class Hierarchy

Pascal

```
TDataSetResolver = class(TCustomResolver);
```

C++

```
class TDataSetResolver : public TCustomResolver;
```

File

Provider

Description

TDataSetProvider (see page 3145) components use TDataSetResolver to apply updates to a dataset when their ResolveToDataSet property is true. Otherwise, they use TSQLResolver (see page 3196).

Do not create instances of TDataSetResolver. Provider (see page 3182) components create their own TDataSetResolver objects for applying updates.

See Also

TDataSetProvider (see page 3145)

TBaseProvider (see page 3077)

TSQLResolver (see page 3196)

58.1.8.1 TDataSetResolver Members

The following tables list the members exposed by TDataSetResolver.

Protected Methods

	Name	Description
	BeginUpdate (see page 3178)	Provides the interface for a method that performs any initialization before applying updates.
	DoDelete (see page 3179)	Defines a method that deletes a record.
	DoInsert (see page 3179)	Defines a method that inserts a record.
	DoUpdate (see page 3180)	Defines a method that modifies a record.
	EndUpdate (see page 3180)	Provides the interface for a method that performs any finalization after applying updates.
	HandleUpdateError (see page 3131)	Manages the resolving process when errors occur while applying updates.
	InitializeConflictBuffer (see page 3181)	Introduces a method for adding information about a failed update that is passed to an OnUpdateError event handler.
	LogUpdateError (see page 3133)	Adds an update that could not be resolved to the error log.
	LogUpdateRecord (see page 3134)	Adds records to the error log if their data has been changed by another application.

Protected Properties

	Name	Description
	Provider (see page 3182)	Indicates the provider that uses this resolver to apply updates.

Public Methods

	Name	Description
	ApplyUpdates (see page 3128)	Applies updates received from a provider.
	Create (see page 3178)	Creates and initializes a dataset resolver.
	Destroy (see page 3127)	Frees an instance of a resolver object.
	FreeTreeData (see page 3130)	Frees the internal data maintained by a TUpdateTree (see page 3205).
	InitKeyFields (see page 3132)	Sets the provider flag pfInKey on all key fields in a delta data packet.
	InitTreeData (see page 3132)	Initializes the internal data maintained by TUpdateTree (see page 3205).

	InternalBeforeResolve (see page 3181)	Initializes internal data structures once the delta packet has been wrapped in an update tree but before any updates are applied.
	InternalUpdateRecord (see page 3133)	Manages the updating process once a delta packet is wrapped in a TUpdateTree (see page 3205) object.
	RowRequest (see page 3134)	Returns information from a specified record.

Public Properties

	Name	Description
	UpdateTree (see page 3135)	This is UpdateTree, a member of class TCustomResolver.

58.1.8.2 TDataSetResolver.Create Constructor

Creates and initializes a dataset resolver.

Pascal

```
constructor Create(AProvider: TDataSetProvider); reintroduce;
```

C++

```
__fastcall TDataSetResolver(TDataSetProvider AProvider);
```

Description

Provider (see page 3182) components call Create when they need a resolver component for applying updates to a dataset. The AProvider parameter specifies the provider that creates and uses the resolver.

Do not create resolver instances directly. Resolvers are designed to be created and used by providers, and can not function independently of a provider.

See Also

Resolver (see page 3088)

58.1.8.3 TDataSetResolver Methods

The methods of the TDataSetResolver class are listed here.

Protected Methods

	Name	Description
	BeginUpdate (see page 3178)	Provides the interface for a method that performs any initialization before applying updates.
	DoDelete (see page 3179)	Defines a method that deletes a record.
	DoInsert (see page 3179)	Defines a method that inserts a record.
	DoUpdate (see page 3180)	Defines a method that modifies a record.
	EndUpdate (see page 3180)	Provides the interface for a method that performs any finalization after applying updates.
	InitializeConflictBuffer (see page 3181)	Introduces a method for adding information about a failed update that is passed to an OnUpdateError event handler.

Public Methods

	Name	Description
	InternalBeforeResolve (see page 3181)	Initializes internal data structures once the delta packet has been wrapped in an update tree but before any updates are applied.

58.1.8.3.1 TDataSetResolver.BeginUpdate Method

Provides the interface for a method that performs any initialization before applying updates.

Pascal

```
procedure BeginUpdate; override;
```

C++

```
virtual __fastcall BeginUpdate();
```

Description

The ApplyUpdates method calls BeginUpdate before applying updates in a delta packet.

As implemented in TCustomResolver, BeginUpdate does nothing. Descendants that need take specific action before applying updates override this method.

See Also

EndUpdate (see page 3130)

ApplyUpdates (see page 3128)

58.1.8.3.2 TDataSetResolver.DoDelete Method

Defines a method that deletes a record.

Pascal

```
procedure DoDelete(Tree: TUpdateTree); override;
```

C++

```
virtual __fastcall DoDelete(TUpdateTree Tree);
```

Description

DoDelete is one of several helper methods internal to the ApplyUpdates method. These helper methods perform the actual updates in the delta packet. ApplyUpdates wraps the delta packet in a TUpdateTree object. For each record that must be deleted, the resolver calls DoDelete.

TCustomResolver does not implement the DoDelete method. Descendants override this method to delete records in their own specialized way.

See Also

ApplyUpdates (see page 3128)

DoInsert (see page 3129)

DoUpdate (see page 3129)

58.1.8.3.3 TDataSetResolver.DoInsert Method

Defines a method that inserts a record.

Pascal

```
procedure DoInsert(Tree: TUpdateTree); override;
```

C++

```
virtual __fastcall DoInsert(TUpdateTree Tree);
```

Description

DoInsert is one of several helper methods internal to the ApplyUpdates method. These helper methods perform the actual updates in the delta packet. ApplyUpdates wraps the delta packet in a TUpdateTree object. For each record that must be

inserted, the resolver calls DoInsert.

TCustomResolver does not implement the DoInsert method. Descendants override this method to insert records in their own specialized way.

See Also

ApplyUpdates (see page 3128)

DoDelete (see page 3129)

DoUpdate (see page 3129)

58.1.8.3.4 TDataSetResolver.DoUpdate Method

Defines a method that modifies a record.

Pascal

```
procedure DoUpdate(Tree: TUpdateTree); override;
```

C++

```
virtual __fastcall DoUpdate(TUpdateTree Tree);
```

Description

DoUpdate is one of several helper methods internal to the ApplyUpdates method. These helper methods perform the actual updates passed in the delta packet. ApplyUpdates wraps the delta packet in a TUpdateTree object. For each record that must be modified, the resolver calls DoUpdate.

TCustomResolver does not implement the DoUpdate method. Descendants override this method to insert records in their own specialized way.

See Also

ApplyUpdates (see page 3128)

DoDelete (see page 3129)

DoInsert (see page 3129)

58.1.8.3.5 TDataSetResolver.EndUpdate Method

Provides the interface for a method that performs any finalization after applying updates.

Pascal

```
procedure EndUpdate; override;
```

C++

```
virtual __fastcall EndUpdate();
```

Description

The ApplyUpdates method calls EndUpdate after applying updates.

As implemented in TCustomResolver, EndUpdate does nothing. Descendants that need take specific action after applying updates override this method.

See Also

BeginUpdate (see page 3128)

ApplyUpdates (see page 3128)

58.1.8.3.6 TDataSetResolver.InitializeConflictBuffer Method

Introduces a method for adding information about a failed update that is passed to an OnUpdateError event handler.

Pascal

```
procedure InitializeConflictBuffer(Tree: TUpdateTree); override;
```

C++

```
virtual __fastcall InitializeConflictBuffer(TUpdateTree Tree);
```

Description

The HandleUpdateError method calls InitializeConflictBuffer before generating an OnUpdateError event. This ensures that the client dataset passed to an OnUpdateError event handler represents the update that caused the problem, with appropriate values for field's CurValue property.

Descendants of TCustomResolver implement this method to obtain information from the dataset or server to which they are applying updates.

See Also

HandleUpdateError (see page 3131)

OnUpdateError (see page 3092)

58.1.8.3.7 TDataSetResolver.InternalBeforeResolve Method

Initializes internal data structures once the delta packet has been wrapped in an update tree but before any updates are applied.

Pascal

```
procedure InternalBeforeResolve(Tree: TUpdateTree); override;
```

C++

```
virtual __fastcall InternalBeforeResolve(TUpdateTree Tree);
```

Description

The ApplyUpdates method wraps delta packets in a TUpdateTree object. Once this is complete, InternalBeforeResolve performs any other initialization before the updates are applied.

As implemented in TCustomResolver, InternalBeforeResolve does nothing. Descendants override this method to perform class-specific initialization.

See Also

ApplyUpdates (see page 3128)

InitTreeData (see page 3132)

58.1.8.4 TDataSetResolver Properties

The properties of the TDataSetResolver class are listed here.

Protected Properties

	Name	Description
	Provider (see page 3182)	Indicates the provider that uses this resolver to apply updates.

58.1.8.4.1 TDataSetResolver.Provider Property

Indicates the provider that uses this resolver to apply updates.

Pascal

```
property Provider: TDataSetProvider;
```

C++

```
__property TDataSetProvider Provider;
```

Description

Provider is a read-only property that points to the provider that uses this resolver for applying updates. The Provider property is set in the constructor.

Resolvers use the Provider property internally to obtain information supplied in the provider's properties. They also use it to generate events on the provider at appropriate points in the update process.

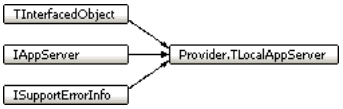
See Also

Create (see page 3126)

Resolver (see page 3088)

58.1.9 TLocalAppServer Class

Class Hierarchy



Pascal

```
TLocalAppServer = class(TInterfacedObject, IAppServer, ISupportErrorInfo);
```

C++

```
class TLocalAppServer : public TInterfacedObject, public IAppServer, public ISupportErrorInfo;
```

File

Provider

Description

This is class Provider.TLocalAppServer.

58.1.9.1 TLocalAppServer Members

The following tables list the members exposed by TLocalAppServer.

Protected Methods

	Name	Description
	AS_ApplyUpdates (see page 3185)	This is AS_ApplyUpdates, a member of class TLocalAppServer.
	AS_DataRequest (see page 3185)	This is AS_DataRequest, a member of class TLocalAppServer.

	AS_Execute (see page 3185)	This is AS_Execute, a member of class TLocalAppServer.
	AS_GetParams (see page 3185)	This is AS_GetParams, a member of class TLocalAppServer.
	AS_GetProviderNames (see page 3186)	This is AS_GetProviderNames, a member of class TLocalAppServer.
	AS_GetRecords (see page 3186)	This is AS_GetRecords, a member of class TLocalAppServer.
	AS_RowRequest (see page 3186)	This is AS_RowRequest, a member of class TLocalAppServer.
	GetIDsOfNames (see page 3186)	This is GetIDsOfNames, a member of class TLocalAppServer.
	GetTypeInfo (see page 3187)	This is GetTypeInfo, a member of class TLocalAppServer.
	GetTypeInfoCount (see page 3187)	This is GetTypeInfoCount, a member of class TLocalAppServer.
	InterfaceSupportsErrorInfo (see page 3187)	This is InterfaceSupportsErrorInfo, a member of class TLocalAppServer.
	Invoke (see page 3187)	This is Invoke, a member of class TLocalAppServer.

Public Methods

	Name	Description
	Create (see page 3183)	This is the overview for the Create constructor overload.
V	Destroy (see page 3184)	Disposes of an object instance.
V	SafeCallException (see page 3188)	Handles exceptions in methods declared using the safecall calling convention.

58.1.9.2 Create Constructor

This is the overview for the Create constructor overload.

Overload List

	Name	Description
	TLocalAppServer.Create (TCustomProvider) (see page 3183)	This is Create, a member of class TLocalAppServer.
	TLocalAppServer.Create (TDataset) (see page 3183)	This is Create, a member of class TLocalAppServer.

58.1.9.2.1 TLocalAppServer.Create Constructor (TCustomProvider)

Pascal

```
constructor Create(AProvider: TCustomProvider); overload;
```

C++

```
__fastcall TLocalAppServer(TCustomProvider AProvider);
```

Description

This is Create, a member of class TLocalAppServer.

58.1.9.2.2 TLocalAppServer.Create Constructor (TDataset)

Pascal

```
constructor Create(ADataset: TDataset); overload;
```

C++

```
__fastcall TLocalAppServer(TDataset ADataset);
```

Description

This is Create, a member of class TLocalAppServer.

58.1.9.3 TLocalAppServer.Destroy Destructor

Disposes of an object instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TLocalAppServer();
```

Description

Do not call System::TObject::Destroy directly. Call System::TObject::Free instead. Free verifies that the object reference is not nil before calling System::TObject::Destroy.

The System::TObject::Destroy method defined by System::TObject deallocates memory. Descendant objects usually define a destructor that is customized for that particular kind of object.

When declaring a System::TObject::Destroy method in a descendant, always add the override directive to the declaration and call the inherited System::TObject::Destroy as the last statement in the overriding method. Since System::TObject::Destroy is a virtual method, overriding it ensures that the proper inherited behavior occurs.

Note: If an exception escapes from the constructor, the destructor is called to destroy the partially constructed object instance that failed to initialize completely. Therefore, destructors should check that allocated resources, such as handles, were actually allocated before trying to release them, since their value might be zero.

System::TObject::Destroy should be implemented so that it calls System::TObject::Free on all subobjects created within the object's constructor (that is, allocated by the constructor). Unlike System::TObject::Destroy, System::TObject::Free provides a safeguard when destroying objects that are nil.

See Also

System::TObject::Free

58.1.9.4 TLocalAppServer Methods

The methods of the TLocalAppServer class are listed here.

Protected Methods

	Name	Description
	AS_ApplyUpdates (see page 3185)	This is AS_ApplyUpdates, a member of class TLocalAppServer.
	AS_DataRequest (see page 3185)	This is AS_DataRequest, a member of class TLocalAppServer.
	AS_Execute (see page 3185)	This is AS_Execute, a member of class TLocalAppServer.
	AS_GetParams (see page 3185)	This is AS_GetParams, a member of class TLocalAppServer.
	AS_GetProviderNames (see page 3186)	This is AS_GetProviderNames, a member of class TLocalAppServer.
	AS_GetRecords (see page 3186)	This is AS_GetRecords, a member of class TLocalAppServer.
	AS_RowRequest (see page 3186)	This is AS_RowRequest, a member of class TLocalAppServer.
	GetIDsOfNames (see page 3186)	This is GetIDsOfNames, a member of class TLocalAppServer.
	GetTypeInfo (see page 3187)	This is GetTypeInfo, a member of class TLocalAppServer.
	GetTypeInfoCount (see page 3187)	This is GetTypeInfoCount, a member of class TLocalAppServer.
	InterfaceSupportsErrorInfo (see page 3187)	This is InterfaceSupportsErrorInfo, a member of class TLocalAppServer.
	Invoke (see page 3187)	This is Invoke, a member of class TLocalAppServer.

Public Methods

	Name	Description
	SafeCallException (see page 3188)	Handles exceptions in methods declared using the safecall calling convention.

58.1.9.4.1 TLocalAppServer.AS_ApplyUpdates Method

Pascal

```
function AS_ApplyUpdates(const ProviderName: WideString; Delta: OleVariant; MaxErrors:
Integer; out ErrorCount: Integer; var OwnerData: OleVariant): OleVariant; safecall;
```

C++

```
__fastcall OleVariant AS_ApplyUpdates(const BSTR ProviderName, OleVariant Delta, int
MaxErrors, int ErrorCount, OleVariant OwnerData);
```

Description

This is AS_ApplyUpdates, a member of class TLocalAppServer.

58.1.9.4.2 TLocalAppServer.AS_DataRequest Method

Pascal

```
function AS_DataRequest(const ProviderName: WideString; Data: OleVariant): OleVariant;
safecall;
```

C++

```
__fastcall OleVariant AS_DataRequest(const BSTR ProviderName, OleVariant Data);
```

Description

This is AS_DataRequest, a member of class TLocalAppServer.

58.1.9.4.3 TLocalAppServer.AS_Execute Method

Pascal

```
procedure AS_Execute(const ProviderName: WideString; const CommandText: WideString; var
Params: OleVariant; var OwnerData: OleVariant); safecall;
```

C++

```
__fastcall AS_Execute(const BSTR ProviderName, const BSTR CommandText, OleVariant Params,
OleVariant OwnerData);
```

Description

This is AS_Execute, a member of class TLocalAppServer.

58.1.9.4.4 TLocalAppServer.AS_GetParams Method

Pascal

```
function AS_GetParams(const ProviderName: WideString; var OwnerData: OleVariant): OleVariant;
safecall;
```

C++

```
__fastcall OleVariant AS_GetParams(const BSTR ProviderName, OleVariant OwnerData);
```

Description

This is AS_GetParams, a member of class TLocalAppServer.

58.1.9.4.5 TLocalAppServer.AS_GetProviderNames Method

Pascal

```
function AS_GetProviderNames: OleVariant; safecall;
```

C++

```
__fastcall OleVariant AS_GetProviderNames();
```

Description

This is AS_GetProviderNames, a member of class TLocalAppServer.

58.1.9.4.6 TLocalAppServer.AS_GetRecords Method

Pascal

```
function AS_GetRecords(const ProviderName: WideString; Count: Integer; out RecsOut: Integer;
Options: Integer; const CommandText: WideString; var Params: OleVariant; var OwnerData:
OleVariant): OleVariant; safecall;
```

C++

```
__fastcall OleVariant AS_GetRecords(const BSTR ProviderName, int Count, int RecsOut, int
Options, const BSTR CommandText, OleVariant Params, OleVariant OwnerData);
```

Description

This is AS_GetRecords, a member of class TLocalAppServer.

58.1.9.4.7 TLocalAppServer.AS_RowRequest Method

Pascal

```
function AS_RowRequest(const ProviderName: WideString; Row: OleVariant; RequestType: Integer;
var OwnerData: OleVariant): OleVariant; safecall;
```

C++

```
__fastcall OleVariant AS_RowRequest(const BSTR ProviderName, OleVariant Row, int RequestType,
OleVariant OwnerData);
```

Description

This is AS_RowRequest, a member of class TLocalAppServer.

58.1.9.4.8 TLocalAppServer.GetIDsOfNames Method

Pascal

```
function GetIDsOfNames(const IID: TGUID; Names: Pointer; NameCount: Integer; LocaleID:
Integer; DispIDs: Pointer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetIDsOfNames(const TGUID IID, void * Names, int NameCount, int
LocaleID, void * DispIDs);
```

Description

This is GetIDsOfNames, a member of class TLocalAppServer.

58.1.9.4.9 TLocalAppServer.GetTypeInfo Method

Pascal

```
function GetTypeInfo(Index: Integer; LocaleID: Integer; out TypeInfo): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetTypeInfo(int Index, int LocaleID,  TypeInfo);
```

Description

This is GetTypeInfo, a member of class TLocalAppServer.

58.1.9.4.10 TLocalAppServer.GetTypeInfoCount Method

Pascal

```
function GetTypeInfoCount(out Count: Integer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT GetTypeInfoCount(int Count);
```

Description

This is GetTypeInfoCount, a member of class TLocalAppServer.

58.1.9.4.11 TLocalAppServer.InterfaceSupportsErrorInfo Method

Pascal

```
function InterfaceSupportsErrorInfo(const iid: TGUID): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT InterfaceSupportsErrorInfo(const TGUID iid);
```

Description

This is InterfaceSupportsErrorInfo, a member of class TLocalAppServer.

58.1.9.4.12 TLocalAppServer.Invoke Method

Pascal

```
function Invoke(DispID: Integer; const IID: TGUID; LocaleID: Integer; Flags: Word; var Params;
VarResult: Pointer; ExcepInfo: Pointer; ArgErr: Pointer): HResult; stdcall;
```

C++

```
__fastcall __stdcall HRESULT Invoke(int DispID, const TGUID IID, int LocaleID, Word Flags,
Params, void * VarResult, void * ExcepInfo, void * ArgErr);
```

Description

This is Invoke, a member of class TLocalAppServer.

58.1.9.4.13 TLocalAppServer.SafeCallException Method

Handles exceptions in methods declared using the safecall calling convention.

Pascal

```
function SafeCallException(ExceptObject: TObject; ExceptAddr: Pointer): HResult; override;
```

C++

```
virtual __fastcall HRESULT SafeCallException(TObject * ExceptObject, void * ExceptAddr);
```

Description

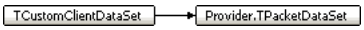
System::TObject::SafeCallException handles exceptions in methods that use the safecall calling convention. Some classes that implement interfaces override this method to handle errors that might occur.

As implemented in System::TObject, System::TObject::SafeCallException simply returns E_UNEXPECTED. This is the appropriate response for classes that do no support interfaces

58.1.10 TPacketDataSet Class

TPacketDataSet represents a data packet or delta packet that is used by a provider.

Class Hierarchy



Pascal

```
TPacketDataSet = class(TCustomClientDataSet);
```

C++

```
class TPacketDataSet : public TCustomClientDataSet;
```

File

Provider

Description

TPacketDataSet represents a data packet that a provider creates to represent its dataset's records and metadata or a delta data packet that a provider receives when it should apply updates. Provider (see page 3073) components create instances of TPacketDataSet to represent data packets before they are sent to clients and to represent delta packets after they are received from clients.

TPacketDataSet provides a convenient way for providers and resolvers to work with data packets. As a descendant of TCustomClientDataSet, TPacketDataSet allows providers and resolvers to use any of the standard dataset properties and methods to manipulate the data packet's records. In addition, TPacketDataSet introduces new properties and methods that describe what information goes into the data packet or how updates should be applied.

Applications do not need to work directly with packet datasets. They are mainly of use to component writers who want to create custom provider components.

See Also

TCustomClientDataSet

TCustomProvider (see page 3096)

58.1.10.1 TPacketDataSet Members

The following tables list the members exposed by TPacketDataSet.

Protected Methods

	Name	Description
	DataEvent (see page 3191)	This is DataEvent, a member of class TPacketDataSet.
	GetStateFieldValue (see page 3191)	This is GetStateFieldValue, a member of class TPacketDataSet.
	InternalClose (see page 3193)	This is InternalClose, a member of class TPacketDataSet.
	InternalInitRecord (see page 3193)	This is InternalInitRecord, a member of class TPacketDataSet.
	InternalOpen (see page 3193)	This is InternalOpen, a member of class TPacketDataSet.
	SetFieldData (see page 3194)	This is SetFieldData, a member of class TPacketDataSet.
	SetStateFieldValue (see page 3194)	This is SetStateFieldValue, a member of class TPacketDataSet.

Protected Properties

	Name	Description
	WritingCurValues (see page 3196)	This is WritingCurValues, a member of class TPacketDataSet.

Public Methods

	Name	Description
	AssignCurValues (see page 3190)	This is the overview for the AssignCurValues method overload.
	Create (see page 3189)	Creates and initializes instance of TPacketDataSet (see page 3188).
	CreateFromDelta (see page 3191)	Copies the data from another packet data set.
	HasCurValues (see page 3192)	Indicates whether the CurValue property is available for fields in the packet dataset.
	HasMergeConflicts (see page 3192)	Indicates whether a current record can be merged with updates in a delta packet.
	InitAltRecBuffers (see page 3192)	Fetches values for the CurValue, OldValue, and NewValue properties of fields.
	UpdateKind (see page 3194)	Indicates the type of update the current record represents.

Public Properties

	Name	Description
	NewValuesModified (see page 3195)	Indicates whether the NewValue property of any field in the current record has changed.
	StreamMetaData (see page 3195)	Indicates whether metadata should be included in the data packet.
	UseCurValues (see page 3195)	Indicates that the CurValue property of fields in the dataset should be used instead of the OldValue property.

58.1.10.2 TPacketDataSet.Create Constructor

Creates and initializes instance of TPacketDataSet (see page 3188).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TPacketDataSet(TComponent * AOwner);
```

Description

Most applications do not need to create packet datasets because they are created by the provider components that use them.Create initializes the properties of the packet dataset.

AOwner is the component that is responsible for freeing the TPacketDataSet (see page 3188) instance. If AOwner is nil (Delphi) or NULL (C++), the application must explicitly free the packet data set object.

See Also

Owner

58.1.10.3 TPacketDataSet Methods

The methods of the TPacketDataSet class are listed here.

Protected Methods

	Name	Description
	DataEvent (see page 3191)	This is DataEvent, a member of class TPacketDataSet.
	GetStateFieldValue (see page 3191)	This is GetStateFieldValue, a member of class TPacketDataSet.
	InternalClose (see page 3193)	This is InternalClose, a member of class TPacketDataSet.
	InternalInitRecord (see page 3193)	This is InternalInitRecord, a member of class TPacketDataSet.
	InternalOpen (see page 3193)	This is InternalOpen, a member of class TPacketDataSet.
	SetFieldData (see page 3194)	This is SetFieldData, a member of class TPacketDataSet.
	SetStateFieldValue (see page 3194)	This is SetStateFieldValue, a member of class TPacketDataSet.

Public Methods

	Name	Description
	AssignCurValues (see page 3190)	This is the overview for the AssignCurValues method overload.
	CreateFromDelta (see page 3191)	Copies the data from another packet data set.
	HasCurValues (see page 3192)	Indicates whether the CurValue property is available for fields in the packet dataset.
	HasMergeConflicts (see page 3192)	Indicates whether a current record can be merged with updates in a delta packet.
	InitAltRecBuffers (see page 3192)	Fetches values for the CurValue, OldValue, and NewValue properties of fields.
	UpdateKind (see page 3194)	Indicates the type of update the current record represents.

58.1.10.3.1 AssignCurValues Method

This is the overview for the AssignCurValues method overload.

Overload List

	Name	Description
	TPacketDataSet.AssignCurValues (TDataSet) (see page 3190)	Updates the current field value for each field in the packet dataset.
	TPacketDataSet.AssignCurValues (Variant) (see page 3190)	Updates the current field value for each field in the packet dataset.

58.1.10.3.1.1 TPacketDataSet.AssignCurValues Method (TDataSet)

Updates the current field value for each field in the packet dataset.

Pascal

```
procedure AssignCurValues(Source: TDataSet); overload;
procedure AssignCurValues(const CurValues: Variant); overload;
```

C++

```
__fastcall AssignCurValues(TDataSet Source);
__fastcall AssignCurValues(const Variant CurValues);
```

Description

When a resolver component encounters an update error, it calls AssignCurValues to set the CurValue property of fields before

triggering the OnUpdateError event handler.

If AssignCurValues is passed a dataset as the Source parameter, then for each field in the packet dataset, the CurValue property is set to the value of the field with the same name in the source dataset.

If AssignCurValues is passed a Variant array as the CurValues parameter, then each entry in the array is used to assign the CurValue property of a field in the packet dataset. CurValues can have one of two forms: an array of field values, or an array of Variant arrays, where each of those arrays has two elements: a field name and a value. In the first form, field values are assigned by position. In the second form, fields are located based on name and the CurValue property assigned accordingly.

See Also

CurValue

OnUpdateError (see page 3092)

58.1.10.3.2 TPacketDataSet.CreateFromDelta Method

Copies the data from another packet data set.

Pascal

```
procedure CreateFromDelta(Source: TPacketDataSet);
```

C++

```
__fastcall CreateFromDelta(TPacketDataSet Source);
```

Description

Applications should not need to call CreateFromDelta. This method is used internally when applying updates to create the error log of updates that could not be resolved. This log of update errors is eventually returned by the provider's ApplyUpdates method.

See Also

ApplyUpdates (see page 3100)

58.1.10.3.3 TPacketDataSet.DataEvent Method

Pascal

```
procedure DataEvent(Event: TDataEvent; Info: Longint); override;
```

C++

```
virtual __fastcall DataEvent(TDataEvent Event, Longint Info);
```

Description

This is DataEvent, a member of class TPacketDataSet.

58.1.10.3.4 TPacketDataSet.GetStateFieldValue Method

Pascal

```
function GetStateFieldValue(State: TDataSetState; Field: TField): Variant; override;
```

C++

```
virtual __fastcall Variant GetStateFieldValue(TDataSetState State, TField Field);
```

Description

This is GetStateFieldValue, a member of class TPacketDataSet.

58.1.10.3.5 TPacketDataSet.HasCurValues Method

Indicates whether the CurValue property is available for fields in the packet dataset.

Pascal

```
function HasCurValues: Boolean;
```

C++

```
__fastcall Boolean HasCurValues();
```

Description

When an error occurs while applying an updated record, resolver components attempt to locate the current record in the underlying database (or the dataset to which updates are applied.) This current record can then be used when trying to determine the cause of the update error. If the current record can be located, HasCurValues returns true and the fields in the packet dataset have their CurValue properties set from this record. If the record can't be located (for example, if another user changed a key field value or deleted the record), then HasCurValues returns false and the CurValue property of packet dataset fields can't be used.

See Also

UseCurValues (see page 3195)

CurValue

58.1.10.3.6 TPacketDataSet.HasMergeConflicts Method

Indicates whether a current record can be merged with updates in a delta packet.

Pascal

```
function HasMergeConflicts: Boolean;
```

C++

```
__fastcall Boolean HasMergeConflicts();
```

Description

When resolving update errors in an OnUpdateError event handler, applications can specify that the updated record should be merged automatically with the current field values of the record. (To do this, set the Response parameter to rrMerge.) However, it is not always possible to merge client edits with the current record. If the updated record includes changes to field values that were also changed by other users, the merge will fail and HasMergeConflicts returns true.

See Also

CurValue

NewValue

OnUpdateError (see page 3092)

58.1.10.3.7 TPacketDataSet.InitAltRecBuffers Method

Fetches values for the CurValue, OldValue, and NewValue properties of fields.

Pascal

```
procedure InitAltRecBuffers(CheckModified: Boolean = True);
```

C++

```
__fastcall InitAltRecBuffers(Boolean CheckModified = True);
```

Description

While applying updates, InitAltRecBuffers is called automatically to ensure that fields in the packet dataset have values for the CurValue, OldValue, and NewValue properties. The CheckModified parameter specifies whether InitAltRecBuffers should check if the current record has been modified when it sets values. This check allows the packet dataset to reflect appropriate values when applying updates that are changed by an OnUpdateError event handler.

See Also

OldValue

NewValue

CurValue

OnUpdateError (see page 3092)

58.1.10.3.8 TPacketDataSet.InternalClose Method

Pascal

```
procedure InternalClose; override;
```

C++

```
virtual __fastcall InternalClose();
```

Description

This is InternalClose, a member of class TPacketDataSet.

58.1.10.3.9 TPacketDataSet.InternalInitRecord Method

Pascal

```
procedure InternalInitRecord(Buffer: TRecordBuffer); override;
```

C++

```
virtual __fastcall InternalInitRecord(TRecordBuffer Buffer);
```

Description

This is InternalInitRecord, a member of class TPacketDataSet.

58.1.10.3.10 TPacketDataSet.InternalOpen Method

Pascal

```
procedure InternalOpen; override;
```

C++

```
virtual __fastcall InternalOpen();
```

Description

This is InternalOpen, a member of class TPacketDataSet.

58.1.10.3.11 TPacketDataSet.SetFieldData Method

Pascal

```
procedure SetFieldData(Field: TField; Buffer: Pointer); override;
```

C++

```
virtual __fastcall SetFieldData(TField Field, void * Buffer);
```

Description

This is SetFieldData, a member of class TPacketDataSet.

58.1.10.3.12 TPacketDataSet.SetStateFieldValue Method

Pascal

```
procedure SetStateFieldValue(State: TDataSetState; Field: TField; const Value: Variant);
override;
```

C++

```
virtual __fastcall SetStateFieldValue(TDataSetState State, TField Field, const Variant Value);
```

Description

This is SetStateFieldValue, a member of class TPacketDataSet.

58.1.10.3.13 TPacketDataSet.UpdateKind Method

Indicates the type of update the current record represents.

Pascal

```
function UpdateKind: TUpdateKind;
```

C++

```
__fastcall TUpdateKind UpdateKind();
```

Description

Call UpdateKind while applying updates to determine the type of update the current record represents. UpdateKind returns

ukInsert if the current record is a newly inserted record.

ukDelete if the current record represents a record that should be deleted.

ukModify if the current record represents updates to an existing record.

58.1.10.4 TPacketDataSet Properties

The properties of the TPacketDataSet class are listed here.

Protected Properties

	Name	Description
	WritingCurValues (see page 3196)	This is WritingCurValues, a member of class TPacketDataSet.

Public Properties

	Name	Description
	NewValuesModified (see page 3195)	Indicates whether the NewValue property of any field in the current record has changed.
	StreamMetaData (see page 3195)	Indicates whether metadata should be included in the data packet.
	UseCurValues (see page 3195)	Indicates that the CurValue property of fields in the dataset should be used instead of the OldValue property.

58.1.10.4.1 TPacketDataSet.NewValuesModified Property

Indicates whether the NewValue property of any field in the current record has changed.

Pascal

```
property NewValuesModified: Boolean;
```

C++

```
__property Boolean NewValuesModified;
```

Description

Read NewValuesModified to determine whether any fields in the current record were edited during the resolving process.

See Also

NewValue

58.1.10.4.2 TPacketDataSet.StreamMetaData Property

Indicates whether metadata should be included in the data packet.

Pascal

```
property StreamMetaData: Boolean;
```

C++

```
__property Boolean StreamMetaData;
```

Description

StreamMetaData reflects the MetaData parameter of the provider's CreateDataPacket method. When StreamMetaData is true, metadata is included in the data packet. When StreamMetaData is false, the data packet contains records but no metadata.

See Also

CreateDataPacket (see page 3149)

58.1.10.4.3 TPacketDataSet.UseCurValues Property

Indicates that the CurValue property of fields in the dataset should be used instead of the OldValue property.

Pascal

```
property UseCurValues: Boolean;
```

C++

```
__property Boolean UseCurValues;
```

Description

UseCurValues is set during the resolving process when a record is corrected in an OnUpdateError event handler. This allows the resolver to reapply a corrected record rather than using the originally provided values as the basis of an update.

See Also

HasCurValues (see page 3192)

CurValue

OldValue

OnUpdateError (see page 3092)

58.1.10.4.4 TPacketDataSet.WritingCurValues Property

Pascal

```
property WritingCurValues: Boolean;
```

C++

```
__property Boolean WritingCurValues;
```

Description

This is WritingCurValues, a member of class TPacketDataSet.

58.1.11 TProvider Class

TProvider provides data to and applies updates from a client dataset.

Class Hierarchy

Pascal

```
TProvider = class(TDataSetProvider);
```

C++

```
class TProvider : public TDataSetProvider;
```

File

Provider

Description

TProvider is provided for backward compatibility. It is identical to TDataSetProvider (see page 3145). New applications should use TDataSetProvider (see page 3145) instead.

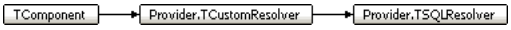
See Also

TDataSetProvider (see page 3145)

58.1.12 TSQLResolver Class

TSQLResolver applies updates to a database server on behalf of a provider.

Class Hierarchy



Pascal

```
TSQLResolver = class(TCustomResolver);
```

C++

```
class TSQLResolver : public TCustomResolver;
```

File

Provider

Description

Provider (see page 3204) components use TSQLResolver to apply updates from a client application directly to a database server when their ResolveToDataSet property is false. Otherwise, they use TDataSetResolver (see page 3176).

Do not create instances of TSQLResolver. Provider (see page 3204) components create their own TSQLResolver objects for applying updates.

See Also

TBaseProvider (see page 3077)

TDataSetResolver (see page 3176)

TDataSetProvider (see page 3145)

58.1.12.1 TSQLResolver Members

The following tables list the members exposed by TSQLResolver.

Protected Methods

	Name	Description
	BeginUpdate (see page 3128)	Provides the interface for a method that performs any initialization before applying updates.
	DoDelete (see page 3199)	Defines a method that deletes a record.
	DoExecSQL (see page 3200)	This is DoExecSQL, a member of class TSQLResolver.
	DoGetValues (see page 3200)	This is DoGetValues, a member of class TSQLResolver.
	DoInsert (see page 3200)	Defines a method that inserts a record.
	DoUpdate (see page 3201)	Defines a method that modifies a record.
	EndUpdate (see page 3130)	Provides the interface for a method that performs any finalization after applying updates.
	GenDeleteSQL (see page 3201)	This is GenDeleteSQL, a member of class TSQLResolver.
	GenInsertSQL (see page 3202)	This is GenInsertSQL, a member of class TSQLResolver.
	GenSelectSQL (see page 3202)	This is GenSelectSQL, a member of class TSQLResolver.
	GenUpdateSQL (see page 3202)	This is GenUpdateSQL, a member of class TSQLResolver.
	GenWhereSQL (see page 3202)	This is GenWhereSQL, a member of class TSQLResolver.
	GetProvider (see page 3203)	This is GetProvider, a member of class TSQLResolver.
	HandleUpdateError (see page 3131)	Manages the resolving process when errors occur while applying updates.
	InitializeConflictBuffer (see page 3203)	Introduces a method for adding information about a failed update that is passed to an OnUpdateError event handler.
	InternalDoUpdate (see page 3204)	This is InternalDoUpdate, a member of class TSQLResolver.
	LogUpdateError (see page 3133)	Adds an update that could not be resolved to the error log.
	LogUpdateRecord (see page 3134)	Adds records to the error log if their data has been changed by another application.
	UseFieldInUpdate (see page 3204)	This is UseFieldInUpdate, a member of class TSQLResolver.
	UseFieldInWhere (see page 3204)	This is UseFieldInWhere, a member of class TSQLResolver.

Protected Properties

	Name	Description
	Provider (see page 3204)	Indicates the provider that uses this resolver to apply updates.

Public Methods

	Name	Description
	ApplyUpdates (see page 3128)	Applies updates received from a provider.
	Create (see page 3198)	Creates and initializes an SQL resolver.
	Destroy (see page 3198)	Destroys an instance of TSQLResolver (see page 3196).
	FreeTreeData (see page 3201)	Frees the internal data maintained by a TUpdateTree.
	InitKeyFields (see page 3132)	Sets the provider flag pfInKey on all key fields in a delta data packet.
	InitTreeData (see page 3203)	Initializes the internal data maintained by TUpdateTree.
	InternalBeforeResolve (see page 3133)	Initializes internal data structures once the delta packet has been wrapped in an update tree but before any updates are applied.
	InternalUpdateRecord (see page 3133)	Manages the updating process once a delta packet is wrapped in a TUpdateTree (see page 3205) object.
	RowRequest (see page 3134)	Returns information from a specified record.

Public Properties

	Name	Description
	UpdateTree (see page 3135)	This is UpdateTree, a member of class TCustomResolver.

58.1.12.2 TSQLResolver.Create Constructor

Creates and initializes an SQL resolver.

Pascal

```
constructor Create(AProvider: TDataSetProvider); reintroduce;
```

C++

```
__fastcall TSQLResolver(TDataSetProvider AProvider);
```

Description

Provider (see page 3204) components call Create when they need a resolver component for applying updates to a database server. Do not create resolver instances directly. Resolvers are designed to be created and used by providers, and can not function independently of a provider.

The AProvider parameter specifies the provider that creates and uses the resolver.

See Also

Resolver (see page 3088)

58.1.12.3 TSQLResolver.Destroy Destructor

Destroys an instance of TSQLResolver (see page 3196).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TSQLResolver();
```

Description

Most applications should not delete instances of TSQLResolver (see page 3196). SQL resolvers are freed automatically by the provider components that use them.

See Also

System::TObject::Free

Resolver (see page 3088)

58.1.12.4 TSQLResolver Methods

The methods of the TSQLResolver class are listed here.

Protected Methods

	Name	Description
	DoDelete (see page 3199)	Defines a method that deletes a record.
	DoExecSQL (see page 3200)	This is DoExecSQL, a member of class TSQLResolver.
	DoGetValues (see page 3200)	This is DoGetValues, a member of class TSQLResolver.
	DoInsert (see page 3200)	Defines a method that inserts a record.
	DoUpdate (see page 3201)	Defines a method that modifies a record.
	GenDeleteSQL (see page 3201)	This is GenDeleteSQL, a member of class TSQLResolver.
	GenInsertSQL (see page 3202)	This is GenInsertSQL, a member of class TSQLResolver.
	GenSelectSQL (see page 3202)	This is GenSelectSQL, a member of class TSQLResolver.
	GenUpdateSQL (see page 3202)	This is GenUpdateSQL, a member of class TSQLResolver.
	GenWhereSQL (see page 3202)	This is GenWhereSQL, a member of class TSQLResolver.
	GetProvider (see page 3203)	This is GetProvider, a member of class TSQLResolver.
	InitializeConflictBuffer (see page 3203)	Introduces a method for adding information about a failed update that is passed to an OnUpdateError event handler.
	InternalDoUpdate (see page 3204)	This is InternalDoUpdate, a member of class TSQLResolver.
	UseFieldInUpdate (see page 3204)	This is UseFieldInUpdate, a member of class TSQLResolver.
	UseFieldInWhere (see page 3204)	This is UseFieldInWhere, a member of class TSQLResolver.

Public Methods

	Name	Description
	FreeTreeData (see page 3201)	Frees the internal data maintained by a TUpdateTree.
	InitTreeData (see page 3203)	Initializes the internal data maintained by TUpdateTree.

58.1.12.4.1 TSQLResolver.DoDelete Method

Defines a method that deletes a record.

Pascal

```
procedure DoDelete(Tree: TUpdateTree); override;
```

C++

```
virtual __fastcall DoDelete(TUpdateTree Tree);
```

Description

DoDelete is one of several helper methods internal to the ApplyUpdates method. These helper methods perform the actual updates in the delta packet. ApplyUpdates wraps the delta packet in a TUpdateTree object. For each record that must be deleted, the resolver calls DoDelete.

TCustomResolver does not implement the DoDelete method. Descendants override this method to delete records in their own

specialized way.

See Also

ApplyUpdates (see page 3128)

DoInsert (see page 3129)

DoUpdate (see page 3129)

58.1.12.4.2 TSQLResolver.DoExecSQL Method

Pascal

```
procedure DoExecSQL(SQL: TWideStringList; Params: TParams); virtual;
```

C++

```
virtual __fastcall DoExecSQL(TWideStringList SQL, TParams Params);
```

Description

This is DoExecSQL, a member of class TSQLResolver.

58.1.12.4.3 TSQLResolver.DoGetValues Method

Pascal

```
procedure DoGetValues(SQL: TWideStringList; Params: TParams; DataSet: TDataSet); virtual;
```

C++

```
virtual __fastcall DoGetValues(TWideStringList SQL, TParams Params, TDataSet DataSet);
```

Description

This is DoGetValues, a member of class TSQLResolver.

58.1.12.4.4 TSQLResolver.DoInsert Method

Defines a method that inserts a record.

Pascal

```
procedure DoInsert(Tree: TUpdateTree); override;
```

C++

```
virtual __fastcall DoInsert(TUpdateTree Tree);
```

Description

DoInsert is one of several helper methods internal to the ApplyUpdates method. These helper methods perform the actual updates in the delta packet. ApplyUpdates wraps the delta packet in a TUpdateTree object. For each record that must be inserted, the resolver calls DoInsert.

TCustomResolver does not implement the DoInsert method. Descendants override this method to insert records in their own specialized way.

See Also

ApplyUpdates (see page 3128)

DoDelete (see page 3129)

DoUpdate (see page 3129)

58.1.12.4.5 TSQLResolver.DoUpdate Method

Defines a method that modifies a record.

Pascal

```
procedure DoUpdate(Tree: TUpdateTree); override;
```

C++

```
virtual __fastcall DoUpdate(TUpdateTree Tree);
```

Description

DoUpdate is one of several helper methods internal to the ApplyUpdates method. These helper methods perform the actual updates passed in the delta packet. ApplyUpdates wraps the delta packet in a TUpdateTree object. For each record that must be modified, the resolver calls DoUpdate.

TCustomResolver does not implement the DoUpdate method. Descendants override this method to insert records in their own specialized way.

See Also

ApplyUpdates (see page 3128)

DoDelete (see page 3129)

DoInsert (see page 3129)

58.1.12.4.6 TSQLResolver.FreeTreeData Method

Frees the internal data maintained by a TUpdateTree.

Pascal

```
procedure FreeTreeData(Tree: TUpdateTree); override;
```

C++

```
virtual __fastcall FreeTreeData(TUpdateTree Tree);
```

Description

The ApplyUpdates method wraps delta packets in a TUpdateTree object. The TUpdateTree stores the delta packet, along with information that the resolver uses when applying updates. This information varies, depending on the class of resolver.

FreeTreeData is called internally just before the resolver's TUpdateTree object is freed. As implemented in TCustomResolver, FreeTreeData does nothing. Descendants override this method if any specific action is required to free the resolver-specific information that is stored in the update tree.

See Also

ApplyUpdates (see page 3128)

InitTreeData (see page 3132)

58.1.12.4.7 TSQLResolver.GenDeleteSQL Method

Pascal

```
procedure GenDeleteSQL(Tree: TUpdateTree; SQL: TWideStrings; Params: TParams; Alias: string);
virtual;
```

C++

```
virtual __fastcall GenDeleteSQL(TUpdateTree Tree, TWideStrings SQL, TParams Params,
UnicodeString Alias);
```

Description

This is GenDeleteSQL, a member of class TSQLResolver.

58.1.12.4.8 TSQLResolver.GenInsertSQL Method

Pascal

```
procedure GenInsertSQL(Tree: TUpdateTree; SQL: TWideStrings; Params: TParams); virtual;
```

C++

```
virtual __fastcall GenInsertSQL(TUpdateTree Tree, TWideStrings SQL, TParams Params);
```

Description

This is GenInsertSQL, a member of class TSQLResolver.

58.1.12.4.9 TSQLResolver.GenSelectSQL Method

Pascal

```
procedure GenSelectSQL(Tree: TUpdateTree; SQL: TWideStrings; Params: TParams; Alias: string;
Mode: TUpdateMode = upWhereKeyOnly); virtual;
```

C++

```
virtual __fastcall GenSelectSQL(TUpdateTree Tree, TWideStrings SQL, TParams Params,
UnicodeString Alias, TUpdateMode Mode = upWhereKeyOnly);
```

Description

This is GenSelectSQL, a member of class TSQLResolver.

58.1.12.4.10 TSQLResolver.GenUpdateSQL Method

Pascal

```
procedure GenUpdateSQL(Tree: TUpdateTree; SQL: TWideStrings; Params: TParams; Alias: string);
virtual;
```

C++

```
virtual __fastcall GenUpdateSQL(TUpdateTree Tree, TWideStrings SQL, TParams Params,
UnicodeString Alias);
```

Description

This is GenUpdateSQL, a member of class TSQLResolver.

58.1.12.4.11 TSQLResolver.GenWhereSQL Method

Pascal

```
procedure GenWhereSQL(Tree: TUpdateTree; SQL: TWideStrings; Params: TParams; GenUpdateMode:
TUpdateMode; Alias: string); virtual;
```

C++

```
virtual __fastcall GenWhereSQL(TUpdateTree Tree, TWideStrings SQL, TParams Params, TUpdateMode
```

```
GenUpdateMode, UnicodeString Alias);
```

Description

This is GenWhereSQL, a member of class TSQLResolver.

58.1.12.4.12 TSQLResolver.GetProvider Method

Pascal

```
function GetProvider: TDataSetProvider; virtual;
```

C++

```
virtual __fastcall TDataSetProvider GetProvider();
```

Description

This is GetProvider, a member of class TSQLResolver.

58.1.12.4.13 TSQLResolver.InitializeConflictBuffer Method

Introduces a method for adding information about a failed update that is passed to an OnUpdateError event handler.

Pascal

```
procedure InitializeConflictBuffer(Tree: TUpdateTree); override;
```

C++

```
virtual __fastcall InitializeConflictBuffer(TUpdateTree Tree);
```

Description

The HandleUpdateError method calls InitializeConflictBuffer before generating an OnUpdateError event. This ensures that the client dataset passed to an OnUpdateError event handler represents the update that caused the problem, with appropriate values for field's CurValue property.

Descendants of TCustomResolver implement this method to obtain information from the dataset or server to which they are applying updates.

See Also

HandleUpdateError (see page 3131)

OnUpdateError (see page 3092)

58.1.12.4.14 TSQLResolver.InitTreeData Method

Initializes the internal data maintained by TUpdateTree.

Pascal

```
procedure InitTreeData(Tree: TUpdateTree); override;
```

C++

```
virtual __fastcall InitTreeData(TUpdateTree Tree);
```

Description

The ApplyUpdates method wraps delta packets in a TUpdateTree object. The TUpdateTree stores the delta packet, along with information that the resolver uses when applying updates. This information varies, depending on the class of resolver.

InitTreeData is called internally when the resolver's TUpdateTree object is created. As implemented in TCustomResolver, InitTreeData does nothing. Descendants override this method to store any class-specific information that they use when applying

updates.

See Also

ApplyUpdates (see page 3128)

FreeTreeData (see page 3130)

58.1.12.4.15 TSQLResolver.InternalDoUpdate Method

Pascal

```
procedure InternalDoUpdate(Tree: TUpdateTree; UpdateKind: TUpdateKind); virtual;
```

C++

```
virtual __fastcall InternalDoUpdate(TUpdateTree Tree, TUpdateKind UpdateKind);
```

Description

This is InternalDoUpdate, a member of class TSQLResolver.

58.1.12.4.16 TSQLResolver.UseFieldInUpdate Method

Pascal

```
function UseFieldInUpdate(Field: TField): Boolean; virtual;
```

C++

```
virtual __fastcall Boolean UseFieldInUpdate(TField Field);
```

Description

This is UseFieldInUpdate, a member of class TSQLResolver.

58.1.12.4.17 TSQLResolver.UseFieldInWhere Method

Pascal

```
function UseFieldInWhere(Field: TField; Mode: TUpdateMode): Boolean; virtual;
```

C++

```
virtual __fastcall Boolean UseFieldInWhere(TField Field, TUpdateMode Mode);
```

Description

This is UseFieldInWhere, a member of class TSQLResolver.

58.1.12.5 TSQLResolver Properties

The properties of the TSQLResolver class are listed here.

Protected Properties

	Name	Description
	Provider (see page 3204)	Indicates the provider that uses this resolver to apply updates.

58.1.12.5.1 TSQLResolver.Provider Property

Indicates the provider that uses this resolver to apply updates.

Pascal

```
property Provider: TDataSetProvider;
```

C++

```
__property TDataSetProvider Provider;
```

Description

Provider is a read-only property that points to the provider that uses this resolver for applying updates. The Provider property is set in the constructor.

Resolvers use the Provider property internally to obtain information supplied in the provider's properties. They also use it to generate events on the provider at appropriate points in the update process.

See Also

Create (see page 3126)

Resolver (see page 3088)

58.1.13 TUpdateTree Class

TUpdateTree stores information about a delta packet on behalf of a resolver.

Class Hierarchy

Pascal

```
TUpdateTree = class(TObject);
```

C++

```
class TUpdateTree : public TObject;
```

File

Provider

Description

When a client dataset applies its updates to the source dataset or database server, the provider that represents that dataset or server passes the updates to a resolver component. The resolver takes this delta data packet and builds a TUpdateTree to store the updates and information about them in a convenient form. In particular, the TUpdateTree structure allows a resolver to handle master/detail relationships (represented as nested dataset fields in the delta packet) so that updates are applied in the proper order.

Applications can't use TUpdateTree directly. Most of the structured information about the delta packet is represented in protected properties that are only available to the resolver classes in the Provider (see page 3073) unit.

See Also

TCustomResolver (see page 3125)

TCustomClientDataSet

TCustomProvider (see page 3096)

58.1.13.1 TUpdateTree Members

The following tables list the members exposed by TUpdateTree.

Protected Properties

	Name	Description
	DetailList (see page 3210)	This is DetailList, a member of class TUpdateTree.
	Opened (see page 3211)	This is Opened, a member of class TUpdateTree.

Public Methods

	Name	Description
	Clear (see page 3207)	This is Clear, a member of class TUpdateTree.
	Create (see page 3206)	Creates and initializes an instance of TUpdateTree (see page 3205).
	Destroy (see page 3207)	Destroys an instance of TUpdateTree (see page 3205).
	DoUpdates (see page 3207)	This is DoUpdates, a member of class TUpdateTree.
	InitData (see page 3208)	This is InitData, a member of class TUpdateTree.
	InitDelta (see page 3208)	This is the overview for the InitDelta method overload.
	InitErrorPacket (see page 3209)	This is InitErrorPacket, a member of class TUpdateTree.
	RefreshData (see page 3209)	This is RefreshData, a member of class TUpdateTree.

Public Properties

	Name	Description
	Data (see page 3209)	This is Data, a member of class TUpdateTree.
	Delta (see page 3210)	This is Delta, a member of class TUpdateTree.
	DetailCount (see page 3210)	This is DetailCount, a member of class TUpdateTree.
	Details (see page 3210)	This is Details, a member of class TUpdateTree.
	ErrorDS (see page 3211)	This is ErrorDS, a member of class TUpdateTree.
	HasErrors (see page 3211)	This is HasErrors, a member of class TUpdateTree.
	IsNested (see page 3211)	This is IsNested, a member of class TUpdateTree.
	Name (see page 3211)	This is Name, a member of class TUpdateTree.
	Parent (see page 3212)	This is Parent, a member of class TUpdateTree.
	Resolver (see page 3212)	This is Resolver, a member of class TUpdateTree.
	Source (see page 3212)	This is Source, a member of class TUpdateTree.

58.1.13.2 TUpdateTree.Create Constructor

Creates and initializes an instance of TUpdateTree (see page 3205).

Pascal

```
constructor Create(AParent: TUpdateTree; AResolver: TCustomResolver);
```

C++

```
__fastcall TUpdateTree(TUpdateTree AParent, TCustomResolver AResolver);
```

Description

Applications do not need to call the TUpdateTree (see page 3205) constructor. Update trees are created automatically by the resolver components that use them.

The AParent parameter specifies the TUpdateTree (see page 3205) for the updates to a master table if this update tree represents a detail data set. If the TUpdateTree (see page 3205) does not represent the updates to a detail dataset, AParent is nil (Delphi) or NULL (C++).

The AResolver parameter specifies the resolver component that uses this update tree when applying updates. This parameter

must be specified because the update tree calls some of this resolver's methods.

58.1.13.3 TUpdateTree.Destroy Destructor

Destroys an instance of TUpdateTree (see page 3205).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TUpdateTree();
```

Description

Applications do not need to free TUpdateTree (see page 3205) instances. Update trees are freed by the resolver components that use them.

See Also

System::TObject::Free

58.1.13.4 TUpdateTree Methods

The methods of the TUpdateTree class are listed here.

Public Methods

	Name	Description
	Clear (see page 3207)	This is Clear, a member of class TUpdateTree.
	DoUpdates (see page 3207)	This is DoUpdates, a member of class TUpdateTree.
	InitData (see page 3208)	This is InitData, a member of class TUpdateTree.
	InitDelta (see page 3208)	This is the overview for the InitDelta method overload.
	InitErrorPacket (see page 3209)	This is InitErrorPacket, a member of class TUpdateTree.
	RefreshData (see page 3209)	This is RefreshData, a member of class TUpdateTree.

58.1.13.4.1 TUpdateTree.Clear Method

Pascal

```
procedure Clear; virtual;
```

C++

```
virtual __fastcall Clear();
```

Description

This is Clear, a member of class TUpdateTree.

58.1.13.4.2 TUpdateTree.DoUpdates Method

Pascal

```
function DoUpdates: Boolean; virtual;
```

C++

```
virtual __fastcall Boolean DoUpdates();
```

Description

This is DoUpdates, a member of class TUpdateTree.

58.1.13.4.3 TUpdateTree.InitData Method

Pascal

```
procedure InitData(ASource: TDataSet); virtual;
```

C++

```
virtual __fastcall InitData(TDataSet ASource);
```

Description

This is InitData, a member of class TUpdateTree.

58.1.13.4.4 InitDelta Method

This is the overview for the InitDelta method overload.

Overload List

	Name	Description
	TUpdateTree.InitDelta (OleVariant) (see page 3208)	This is InitDelta, a member of class TUpdateTree.
	TUpdateTree.InitDelta (TPacketDataSet) (see page 3208)	This is InitDelta, a member of class TUpdateTree.

58.1.13.4.4.1 TUpdateTree.InitDelta Method (OleVariant)

Pascal

```
procedure InitDelta(const ADelta: OleVariant); virtual; overload;
```

C++

```
virtual __fastcall InitDelta(const OleVariant ADelta);
```

Description

This is InitDelta, a member of class TUpdateTree.

58.1.13.4.4.2 TUpdateTree.InitDelta Method (TPacketDataSet)

Pascal

```
procedure InitDelta(ADelta: TPacketDataSet); virtual; overload;
```

C++

```
virtual __fastcall InitDelta(TPacketDataSet ADelta);
```

Description

This is InitDelta, a member of class TUpdateTree.

58.1.13.4.5 TUpdateTree.InitErrorPacket Method

Pascal

```
procedure InitErrorPacket(E: EUpdateError; Response: TResolverResponse); virtual;
```

C++

```
virtual __fastcall InitErrorPacket(EUpdateError E, TResolverResponse Response);
```

Description

This is InitErrorPacket, a member of class TUpdateTree.

58.1.13.4.6 TUpdateTree.RefreshData Method

Pascal

```
procedure RefreshData(Options: TFetchOptions); virtual;
```

C++

```
virtual __fastcall RefreshData(TFetchOptions Options);
```

Description

This is RefreshData, a member of class TUpdateTree.

58.1.13.5 TUpdateTree Properties

The properties of the TUpdateTree class are listed here.

Protected Properties

	Name	Description
	DetailList (see page 3210)	This is DetailList, a member of class TUpdateTree.
	Opened (see page 3211)	This is Opened, a member of class TUpdateTree.

Public Properties

	Name	Description
	Data (see page 3209)	This is Data, a member of class TUpdateTree.
	Delta (see page 3210)	This is Delta, a member of class TUpdateTree.
	DetailCount (see page 3210)	This is DetailCount, a member of class TUpdateTree.
	Details (see page 3210)	This is Details, a member of class TUpdateTree.
	ErrorDS (see page 3211)	This is ErrorDS, a member of class TUpdateTree.
	HasErrors (see page 3211)	This is HasErrors, a member of class TUpdateTree.
	IsNested (see page 3211)	This is IsNested, a member of class TUpdateTree.
	Name (see page 3211)	This is Name, a member of class TUpdateTree.
	Parent (see page 3212)	This is Parent, a member of class TUpdateTree.
	Resolver (see page 3212)	This is Resolver, a member of class TUpdateTree.
	Source (see page 3212)	This is Source, a member of class TUpdateTree.

58.1.13.5.1 TUpdateTree.Data Property

Pascal

```
property Data: Pointer;
```

C++

```
__property void * Data;
```

Description

This is Data, a member of class TUpdateTree.

58.1.13.5.2 TUpdateTree.Delta Property

Pascal

```
property Delta: TPacketDataSet;
```

C++

```
__property TPacketDataSet Delta;
```

Description

This is Delta, a member of class TUpdateTree.

58.1.13.5.3 TUpdateTree.DetailCount Property

Pascal

```
property DetailCount: Integer;
```

C++

```
__property int DetailCount;
```

Description

This is DetailCount, a member of class TUpdateTree.

58.1.13.5.4 TUpdateTree.DetailList Property

Pascal

```
property DetailList: TList;
```

C++

```
__property TList DetailList;
```

Description

This is DetailList, a member of class TUpdateTree.

58.1.13.5.5 TUpdateTree.Details Property

Pascal

```
property Details [Index: Integer]: TUpdateTree;
```

C++

```
__property TUpdateTree Details[int Index];
```

Description

This is Details, a member of class TUpdateTree.

58.1.13.5.6 TUpdateTree.ErrorDS Property

Pascal

```
property ErrorDS: TPacketDataSet;
```

C++

```
__property TPacketDataSet ErrorDS;
```

Description

This is ErrorDS, a member of class TUpdateTree.

58.1.13.5.7 TUpdateTree.HasErrors Property

Pascal

```
property HasErrors: Boolean;
```

C++

```
__property Boolean HasErrors;
```

Description

This is HasErrors, a member of class TUpdateTree.

58.1.13.5.8 TUpdateTree.IsNested Property

Pascal

```
property IsNested: Boolean;
```

C++

```
__property Boolean IsNested;
```

Description

This is IsNested, a member of class TUpdateTree.

58.1.13.5.9 TUpdateTree.Name Property

Pascal

```
property Name: WideString;
```

C++

```
__property BSTR Name;
```

Description

This is Name, a member of class TUpdateTree.

58.1.13.5.10 TUpdateTree.Opened Property

Pascal

```
property Opened: Boolean;
```

C++

```
__property Boolean Opened;
```

Description

This is Opened, a member of class TUpdateTree.

58.1.13.5.11 TUpdateTree.Parent Property

Pascal

```
property Parent: TUpdateTree;
```

C++

```
__property TUpdateTree Parent;
```

Description

This is Parent, a member of class TUpdateTree.

58.1.13.5.12 TUpdateTree.Resolver Property

Pascal

```
property Resolver: TCustomResolver;
```

C++

```
__property TCustomResolver Resolver;
```

Description

This is Resolver, a member of class TUpdateTree.

58.1.13.5.13 TUpdateTree.Source Property

Pascal

```
property Source: TDataSet;
```

C++

```
__property TDataSet Source;
```

Description

This is Source, a member of class TUpdateTree.

58.2 Interfaces

The following table lists interfaces in this documentation.

Interfaces

	Name	Description
	IProviderContainer (see page 3213)	This is class Provider.IProviderContainer.

58.2.1 IProviderContainer Interface

Class Hierarchy

Provider.IProviderContainer

Pascal

```
IProviderContainer = interface;
```

C++

```
__interface IProviderContainer;
```

File

Provider

Description

This is class Provider.IProviderContainer.

58.2.1.1 IProviderContainer Members

The following tables list the members exposed by IProviderContainer.

Public Methods

	Name	Description
	RegisterProvider (see page 3214)	This is RegisterProvider, a member of class IProviderContainer.
	UnRegisterProvider (see page 3214)	This is UnRegisterProvider, a member of class IProviderContainer.

58.2.1.2 IProviderContainer Methods

The methods of the IProviderContainer class are listed here.

Public Methods

	Name	Description
	RegisterProvider (see page 3214)	This is RegisterProvider, a member of class IProviderContainer.
	UnRegisterProvider (see page 3214)	This is UnRegisterProvider, a member of class IProviderContainer.

58.2.1.2.1 IProviderContainer.RegisterProvider Method

Pascal

```
procedure RegisterProvider(Prov: TCustomProvider);
```

C++

```
__fastcall RegisterProvider(TCustomProvider Prov);
```

Description

This is RegisterProvider, a member of class IProviderContainer.

58.2.1.2.2 IProviderContainer.UnRegisterProvider Method

Pascal

```
procedure UnRegisterProvider(Prov: TCustomProvider);
```

C++

```
__fastcall UnRegisterProvider(TCustomProvider Prov);
```

Description

This is UnRegisterProvider, a member of class IProviderContainer.

58.3 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	GetObjectProperty (see page 3215)	This is function Provider.GetObjectProperty.
	GetStringProperty (see page 3215)	This is function Provider.GetStringProperty.
	VarArrayFromStrings (see page 3215)	Creates a Variant array that contains the strings in a TStrings object.
	VarArrayFromWideStrings (see page 3216)	This is function Provider.VarArrayFromWideStrings.

58.3.1 Provider.GetObjectProperty Function

Pascal

```
function GetObjectProperty(Instance: TPersistent; const PropName: string): TObject;
```

C++

```
TObject * GetObjectProperty(TPersistent * Instance, const UnicodeString PropName);
```

File

Provider

Description

This is function Provider.GetObjectProperty.

58.3.2 Provider.GetStringProperty Function

Pascal

```
function GetStringProperty(Instance: TPersistent; const PropName: string): WideString;
```

C++

```
BSTR GetStringProperty(TPersistent * Instance, const UnicodeString PropName);
```

File

Provider

Description

This is function Provider.GetStringProperty.

58.3.3 Provider.VarArrayFromStrings Function

Creates a Variant array that contains the strings in a TStrings object.

Pascal

```
function VarArrayFromStrings(Strings: TStrings): Variant;
```

C++

```
Variant VarArrayFromStrings(TStrings Strings);
```

File

Provider

Description

Call VarArrayFromStrings to create a Variant array to represent the strings in a TStrings object. VarArrayFromStrings creates a new Variant array of WideString values, where each WideString corresponds to a string in the specified TStrings object.

Strings is the TStrings object that supplies the strings for the Variant array.

VarArrayFromStrings returns the new Variant array.

See Also

PackageParams

VarArrayCreate

58.3.4 Provider.VarArrayFromWideStrings Function

Pascal

```
function VarArrayFromWideStrings(Strings: TWideStrings): Variant;
```

C++

```
Variant VarArrayFromWideStrings(TWideStrings Strings);
```

File

Provider

Description

This is function Provider.VarArrayFromWideStrings.

58.4 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TGetRecordOption (see page 3217)	TGetRecordOption enumerates the types of information that can be added to data packets.
	TProviderOption (see page 3218)	TProviderOption and TProviderOptions (see page 3228) influence the type of information that is communicated between client datasets and provider components.
	TResolverResponse (see page 3221)	TResolverResponse indicates the action a resolver component should take after an OnUpdateError event handler handles an update error.

Records

	Name	Description
	TPutFieldInfo (see page 3220)	TPutFieldInfo contains information about a field.

58.4.1 Provider.TGetRecordOption Enumeration

TGetRecordOption enumerates the types of information that can be added to data packets.

Pascal

```
TGetRecordOption = (
  grMetaData,
  grReset,
  grXML,
  grXMLUTF8
);
```

C++

```
enum TGetRecordOption {
  grMetaData,
  grReset,
  grXML,
  grXMLUTF8
};
```

File

Provider

Description

TGetRecordOption values are listed in the following table.

Value	Meaning
grMetaData	The data packet includes metadata as well as record values.
grReset	The records in the data packet start with the first record, regardless of the contents of any previously sent data packets.
grXML	The data packet is transmitted as a string of XML instead of as an OleVariant. Extended characters are expressed using escape sequences.

grXMLUTF8	The data packet is transmitted as a string of XML instead of as an OleVariant. Extended characters are encoded using UTF-8.

See Also

TCustomProvider (see page 3096)

TGetRecordOptions (see page 3226)

58.4.2 Provider.TProviderOption Enumeration

TProviderOption and TProviderOptions (see page 3228) influence the type of information that is communicated between client datasets and provider components.

Pascal

```
TProviderOption = (
  poFetchBlobsOnDemand,
  poFetchDetailsOnDemand,
  poIncFieldProps,
  poCascadeDeletes,
  poCascadeUpdates,
  poReadOnly,
  poAllowMultiRecordUpdates,
  poDisableInserts,
  poDisableEdits,
  poDisableDeletes,
  poNoReset,
  poAutoRefresh,
  poPropogateChanges,
  poAllowCommandText,
  poRetainServerOrder,
  poUseQuoteChar
);
```

C++

```
enum TProviderOption {
  poFetchBlobsOnDemand,
  poFetchDetailsOnDemand,
  poIncFieldProps,
  poCascadeDeletes,
  poCascadeUpdates,
  poReadOnly,
  poAllowMultiRecordUpdates,
  poDisableInserts,
  poDisableEdits,
  poDisableDeletes,
  poNoReset,
  poAutoRefresh,
  poPropogateChanges,
  poAllowCommandText,
  poRetainServerOrder,
  poUseQuoteChar
};
```

File

Provider

Description

TProviderOptions (see page 3228) is a set of TProviderOption values. TProviderOption values are listed in the following table.

Value	Meaning
poFetchBlobsOnDemand	BLOB fields are not included in data packets. Instead, client applications must request these values on an as-needed basis. If the client dataset's FetchOnDemand property is True, then the client requests these values automatically. Otherwise, the client application uses the FetchBlobs method to retrieve BLOB data.
poFetchDetailsOnDemand	When the provider represents the master of a master/detail relationship, nested detail values are not included in the data packet. Instead, client applications request these on an as-needed basis. If the client dataset's FetchOnDemand property is True, then the client requests these values automatically. Otherwise the client application uses the FetchDetails method to retrieve nested detail sets.
poIncFieldProps	The data packet includes the following field properties (where applicable): Alignment, DisplayLabel, DisplayWidth, Visible, DisplayFormat, EditFormat, MaxValue, MinValue, Currency, EditMask, DisplayValues.
poCascadeDeletes	Tells the server to automatically delete detail records when master table records are deleted. To use this option, the provider must represent the master of a master/detail relationship and the database must support cascaded deletes as part of its referential integrity settings.
poCascadeUpdates	Tells the server to update detail records automatically when master table key values are changed. To use this option, the provider must represent the master of a master/detail relationship and the database must support cascaded updates as part of its referential integrity settings.
poReadOnly	The client dataset can't apply updates to the provider.
poAllowMultiRecordUpdates	Enables individual updates that affect multiple records. When poAllowMultiRecordUpdates is not included, updates that change multiple records are automatically aborted.
poDisableInserts	Clients cannot add new records. If the client tries to insert a record, an exception is raised. (This does not affect the client's ability to edit or delete records.)
poDisableEdits	Clients cannot modify existing data values. If the client tries to edit a field, an exception is raised. (This does not affect the client's ability to insert or delete records.)
poDisableDeletes	Clients cannot delete records. If the client tries to delete a record, an exception is raised. (This does not affect the client's ability to edit or insert records.)
poNoReset	Ignores the reset flag in calls to AS_GetRecords (TGetRecordsOption value of grReset).
poAutoRefresh	Refreshes the client dataset with current record values whenever it applies updates.
poPropogateChanges	Changes made in a BeforeUpdateRecord or AfterUpdateRecord event handler are sent back to the client and merged into the client dataset.
poAllowCommandText	Enables the client to override the provider's dataset by replacing its associated SQL text or the name of its stored procedure or table. These overrides are supplied to the GetRecords or Execute method.
poRetainServerOrder	Signals that the client should not attempt to re-establish the default order by sorting the records in the data packet. This prevents the client from changing the record order because, for example, it is unable to duplicate the locale on the server.
poUseQuoteChar	Specifies that the quotation mark (“) used by the SQL provider will also be used to quote SQL identifiers in the SQL queries created by the provider. This option is True by default; set it to False to eliminate the quotes.

See Also

TBaseProvider (see page 3077)

TGetRecordOption (see page 3217)

58.4.3 Provider.TPutFieldInfo Record

TPutFieldInfo contains information about a field.

Pascal

```
TPutFieldInfo = record
  FieldNo: Integer;
  Field: TField;
  DataSet: TDataSet;
  Size: Integer;
  IsDetail: Boolean;
  Opened: Boolean;
  PutProc: TPutFieldProc;
  LocalFieldIndex: Integer;
  FieldInfos: Pointer;
end;
```

C++

```
struct TPutFieldInfo {
  int FieldNo;
  TField Field;
  TDataSet DataSet;
  int Size;
  Boolean IsDetail;
  Boolean Opened;
  TPutFieldProc PutProc;
  int LocalFieldIndex;
  void * FieldInfos;
};
```

File

Provider

Description

TPutFieldInfo is a record containing information about a field. TPutFieldInfo is used when adding a new field to a data set. TPutFieldInfo fields are listed in the following table.

Field	Meaning
FieldNo	The position of the field (column) in the underlying database table.
Field	The actual field component.
DataSet	Identifies the dataset to which the field component belongs.
Size	Indicates the size used in the definition of the physical database field for data types that support different sizes.
IsDetail	Specifies whether the field is linked to a data set.
Opened	Specifies whether the linked data set is active, in the case when the field is linked to a data set.
PutProc	Stores the procedure used to add the field to the data set.
LocalFieldIndex	Index variable used internally.
FieldInfos	In the case when the field is linked to a data set, FieldInfos contains information about all the fields in the linked data set. To access the information about the fields, do a typecast on FieldInfos to a TInfoArray (see page 3227).

See Also

TInfoArray (see page 3227)

PPutFieldInfo (see page 3222)

58.4.4 Provider.TResolverResponse Enumeration

TResolverResponse indicates the action a resolver component should take after an OnUpdateError event handler handles an update error.

Pascal

```
TResolverResponse = (
  rrSkip,
  rrAbort,
  rrMerge,
  rrApply,
  rrIgnore
);
```

C++

```
enum TResolverResponse {
  rrSkip,
  rrAbort,
  rrMerge,
  rrApply,
  rrIgnore
};
```

File

Provider

Description

The following table lists the possible values for TResolverResponse and what they indicate:

Value	Meaning
rrSkip	Skip updating the record that raised the error condition, and leave the unapplied changes in the cache.
rrAbort	Abort the entire update operation without displaying an error message.
rrMerge	Merge the updates in the delta packet with the new underlying value from the database. This will only succeed if none of the updated fields have been changed by another user.
rrApply	Apply the current record value (as it has been fixed in the OnUpdateError handler) instead of the original update.
rrIgnore	Ignore the error condition: Do not apply the record but do not return the record to the client dataset for resolving either.

See Also

OnUpdateError (see page 3092)

TCustomResolver (see page 3125)

58.5 Types

The following table lists types in this documentation.

Types

Name	Description
PPutFieldInfo (see page 3222)	PPutFieldInfo is a pointer type to a TPutFieldInfo (see page 3220) record.
TAfterUpdateRecordEvent (see page 3223)	TAfterUpdateRecordEvent represents the type of events that occur when the associated resolver has just updated a record.
TBeforeCommitEvent (see page 3223)	TBeforeCommitEvent represents the procedure type for the events that occur before committing a data set.
TBeforeUpdateRecordEvent (see page 3224)	TBeforeUpdateRecordEvent represents the procedure type for the events that occur before a record is updated.
TDataRequestEvent (see page 3224)	Represents the function type for the events that occur when a provider calls its DataRequest method.
TGetDSProps (see page 3225)	Represents the procedure type for the events that occur when the provider adds additional information to a data packet it is creating.
TGetParamsEvent (see page 3225)	Represents the procedure type for the events that occur when the provider adds attributes to the data set.
TGetRecordOptions (see page 3226)	TGetRecordOptions is used to decide what type of information is added to data packets.
TGetTableNameEvent (see page 3226)	Represents the procedure type for the events that occur when a resolver initializes its information about the table to which it applies updates.
TInfoArray (see page 3227)	TInfoArray is an array of TPutFieldInfo (see page 3220) records.
TOnValidate (see page 3227)	Represents the procedure type for the events that occur when the provider applies updates contained in a `Delta` packet.
TProviderDataEvent (see page 3228)	Represents the procedure type for the events that occur when the provider applies updates or retrieves data.
TProviderOptions (see page 3228)	TProviderOptions influences the type of information that is communicated between client datasets and provider components.
TPutFieldProc (see page 3229)	Allows to decide the procedure used when adding a new field to a data set.
TResolverErrorEvent (see page 3230)	Represents the procedure type for the events that occur when the provider fails to update a record.

58.5.1 Provider.PPutFieldInfo Type

PPutFieldInfo is a pointer type to a TPutFieldInfo (see page 3220) record.

Pascal

```
PPutFieldInfo = ^TPutFieldInfo;
```

C++

```
^TPutFieldInfo PPutFieldInfo;
```

File

Provider

Description

Declare a PPutFieldInfo type variable to reference a TPutFieldInfo (see page 3220) record, which contains information used when adding a new field to the data set.

See Also

TPutFieldInfo (see page 3220)

58.5.2 Provider.TAfterUpdateRecordEvent Type

TAfterUpdateRecordEvent represents the type of events that occur when the associated resolver has just updated a record.

Pascal

```
TAfterUpdateRecordEvent = procedure (Sender: TObject; SourceDS: TDataSet; DeltaDS:
TCustomClientDataSet; UpdateKind: TUpdateKind) of object;
```

C++

```
(Sender: TObject; SourceDS: TDataSet; DeltaDS: TCustomClientDataSet; UpdateKind: TUpdateKind)
( TAfterUpdateRecordEvent)();
```

File

Provider

Description

Call TAfterUpdateRecordEvent to provide special processing immediately after a record has been successfully updated. For example, you can use an object of the TAfterUpdateRecordEvent type to maintain a log of successful updates.

See Also

TBeforeUpdateRecordEvent (see page 3224)

Procedural types

Creating events

58.5.3 Provider.TBeforeCommitEvent Type

TBeforeCommitEvent represents the procedure type for the events that occur before committing a data set.

Pascal

```
TBeforeCommitEvent = procedure (Sender:TObject; DeltaDS: TCustomClientDataSet; ErrorCount,
MaxErrors: integer; const ResultVar: OleVariant) of object;
```

C++

```
(Sender:TObject; DeltaDS: TCustomClientDataSet; ErrorCount, MaxErrors: integer; const
ResultVar: OleVariant) ( TBeforeCommitEvent)();
```

File

Provider

Description

A variable of TBeforeCommitEvent type references an event that occurs before the associated resolver commits the changes to the data set.

See Also

TAfterUpdateRecordEvent (see page 3223)

TBeforeUpdateRecordEvent (see page 3224)

Procedural types

Creating events

58.5.4 Provider.TBeforeUpdateRecordEvent Type

TBeforeUpdateRecordEvent represents the procedure type for the events that occur before a record is updated.

Pascal

```
TBeforeUpdateRecordEvent = procedure (Sender: TObject; SourceDS: TDataSet; DeltaDS:
TCustomClientDataSet; UpdateKind: TUpdateKind; var Applied: Boolean) of object;
```

C++

```
(Sender: TObject; SourceDS: TDataSet; DeltaDS: TCustomClientDataSet; UpdateKind: TUpdateKind;
var Applied: Boolean) ( TBeforeUpdateRecordEvent)();
```

File

Provider

Description

A variable of TBeforeUpdateRecordEvent type references an event that occurs before the associated resolver updates a record.

See Also

TAfterUpdateRecordEvent (see page 3223)

Procedural types

Creating events

58.5.5 Provider.TDataRequestEvent Type

Represents the function type for the events that occur when a provider calls its DataRequest method.

Pascal

```
TDataRequestEvent = function (Sender: TObject; Input: OleVariant): OleVariant of object;
```

C++

```
(Sender: TObject; Input: OleVariant): OleVariant ( TDataRequestEvent)();
```

File

Provider

Description

Use a variable of TDataRequestEvent type to reference an event that occurs when a provider calls its DataRequest method.

See Also

DataRequest (see page 3101)

OnDataRequest (see page 3124)

Procedural types

Creating events

58.5.6 Provider.TGetDSProps Type

Represents the procedure type for the events that occur when the provider adds additional information to a data packet it is creating.

Pascal

```
TGetDSProps = procedure (Sender: TObject; DataSet: TDataSet; out Properties: OleVariant) of object;
```

C++

```
(Sender: TObject; DataSet: TDataSet; out Properties: OleVariant) ( TGetDSProps)();
```

File

Provider

Description

A variable of TGetDSProps type references an event that occurs when the provider adds additional information to a data packet it is creating.

See Also

OnGetDataSetProperties (see page 3174)

58.5.7 Provider.TGetParamsEvent Type

Represents the procedure type for the events that occur when the provider adds attributes to the data set.

Pascal

```
TGetParamsEvent = procedure (DataSet: TDataSet; Params: TList) of object;
```

C++

```
(DataSet: TDataSet; Params: TList) ( TGetParamsEvent)();
```

File

Provider

Description

A variable of TGetParamsEvent type references an event that occurs when the provider adds attributes to the data set.

See Also

AddDataSetAttributes (see page 3139)

Procedural types

Creating events

58.5.8 Provider.TGetRecordOptions Type

TGetRecordOptions is used to decide what type of information is added to data packets.

Pascal

```
TGetRecordOptions = set of TGetRecordOption;
```

C++

```
TGetRecordOption TGetRecordOptions;
```

File

Provider

Description

TGetRecordOptions is a set of TGetRecordOption (see page 3217) values. TGetRecordOption (see page 3217) values are listed in the following table.

Value	Meaning
grMetaData	The data packet includes metadata as well as record values.
grReset	The records in the data packet start with the first record, regardless of the contents of any previously sent data packets.
grXML	The data packet is transmitted as a string of XML instead of as an OleVariant. Extended characters are expressed using escape sequences.
grXMLUTF8	The data packet is transmitted as a string of XML instead of as an OleVariant. Extended characters are encoded using UTF-8.

See Also

TCustomProvider (see page 3096)

58.5.9 Provider.TGetTableNameEvent Type

Represents the procedure type for the events that occur when a resolver initializes its information about the table to which it applies updates.

Pascal

```
TGetTableNameEvent = procedure (Sender: TObject; DataSet: TDataSet; var TableName: WideString)
of object;
```

C++

```
(Sender: TObject; DataSet: TDataSet; var TableName: WideString) ( TGetTableNameEvent)();
```

File

Provider

Description

A variable of TGetTableNameEvent type references an event that enables the provider to apply updates to datasets where the resolver cannot determine the target table in the provider's dataset.

See Also

OnGetTableName (see page 3175)

Procedural types

Creating events

58.5.10 Provider.TInfoArray Type

TInfoArray is an array of TPutFieldInfo (see page 3220) records.

Pascal

```
TInfoArray = array of TPutFieldInfo;
```

C++

```
array of TPutFieldInfo TInfoArray;
```

File

Provider

Description

Use a variable of TInfoArray type to store information about adding an entire record to a data set. The record contains several fields of various types.

See Also

TPutFieldInfo (see page 3220)

PPutFieldInfo (see page 3222)

58.5.11 Provider.TOnValidate Type

Represents the procedure type for the events that occur when the provider applies updates contained in a `Delta` packet.

Pascal

```
TOnValidate = procedure (const Delta: OleVariant) of object;
```

C++

```
(const Delta: OleVariant) ( TOnValidate)();
```

File

Provider

Description

A variable of TOnValidate type references an event that occurs when the provider applies updates contained in a `Delta` packet.

See Also

OnValidate (see page 3125)

58.5.12 Provider.TProviderDataEvent Type

Represents the procedure type for the events that occur when the provider applies updates or retrieves data.

Pascal

```
TProviderDataEvent = procedure (Sender: TObject; DataSet: TCustomClientDataSet) of object;
```

C++

```
(Sender: TObject; DataSet: TCustomClientDataSet) ( TProviderDataEvent)();
```

File

Provider

Description

A variable of TProviderDataEvent type references an event that occurs either when the provider begins to apply updates or when the provider retrieves data.

See Also

OnUpdateData (see page 3092)

OnGetData (see page 3091)

Procedural types

Creating events

58.5.13 Provider.TProviderOptions Type

TProviderOptions influences the type of information that is communicated between client datasets and provider components.

Pascal

```
TProviderOptions = set of TProviderOption;
```

C++

```
TProviderOption TProviderOptions;
```

File

Provider

Description

TProviderOptions is a set of TProviderOption (see page 3218) values.

TProviderOption (see page 3218) values are listed in the following table.

Value	Meaning
poFetchBlobsOnDemand	BLOB fields are not included in data packets. Instead, client applications must request these values on an as-needed basis. If the client dataset's FetchOnDemand property is True, then the client requests these values automatically. Otherwise, the client application uses the FetchBlobs method to retrieve BLOB data.

poFetchDetailsOnDemand	When the provider represents the master of a master/detail relationship, nested detail values are not included in the data packet. Instead, client applications request these on an as-needed basis. If the client dataset's FetchOnDemand property is True, then the client requests these values automatically. Otherwise the client application uses the FetchDetails method to retrieve nested detail sets.
poIncFieldProps	The data packet includes the following field properties (where applicable): Alignment, DisplayLabel, DisplayWidth, Visible, DisplayFormat, EditFormat, MaxValue, MinValue, Currency, EditMask, DisplayValues.
poCascadeDeletes	Tells the server to delete detail records automatically when master table records are deleted. To use this option, the provider must represent the master of a master/detail relationship and the database must support cascaded deletes as part of its referential integrity settings.
poCascadeUpdates	Tells the server to automatically update detail records when master table key values are changed. To use this option, the provider must represent the master of a master/detail relationship and the database must support cascaded updates as part of its referential integrity settings.
poReadOnly	The client dataset cannot apply updates to the provider.
poAllowMultiRecordUpdates	Enables individual updates that affect multiple records. When poAllowMultiRecordUpdates is not included, updates that change multiple records are automatically aborted.
poDisableInserts	Clients cannot add new records. If the client tries to insert a record, an exception is raised. (This does not affect the client's ability to edit or delete records.)
poDisableEdits	Clients cannot modify existing data values. If the client tries to edit a field, an exception is raised. (This does not affect the client's ability to insert or delete records.)
poDisableDeletes	Clients cannot delete records. If the client tries to delete a record, an exception is raised. (This does not affect the client's ability to edit or insert records.)
poNoReset	Ignores the reset flag in calls to AS_GetRecords (TGetRecordsOption value of grReset).
poAutoRefresh	Refreshes the client dataset with current record values whenever it applies updates.
poPropogateChanges	Changes made in a BeforeUpdateRecord or AfterUpdateRecord event handler are sent back to the client and merged into the client dataset.
poAllowCommandText	Enables the client to override the provider's dataset by replacing its associated SQL text or the name of its stored procedure or table. These overrides are supplied to the GetRecords or Execute method.
poRetainServerOrder	Signals that the client should not attempt to re-establish the default order by sorting the records in the data packet. This prevents the client from changing the record order because, for example, it is unable to duplicate the locale on the server.
poUseQuoteChar	Specifies that the quotation mark (“) used by the SQL provider will also be used to quote SQL identifiers in the SQL queries created by the provider. This option is True by default; set it to False to eliminate the quotes.

See Also

TBaseProvider (see page 3077)

TGetRecordOption (see page 3217)

58.5.14 Provider.TPutFieldProc Type

Allows to decide the procedure used when adding a new field to a data set.

Pascal

```
TPutFieldProc = procedure (Info: PPutFieldInfo) of object;
```

C++

```
(Info: PPutFieldInfo) ( TPutFieldProc)();
```

File

Provider

Description

A variable of the TPutFieldProc type references a procedure used when adding a new field to a data set.

See Also

TPutFieldInfo (see page 3220)

PPutFieldInfo (see page 3222)

58.5.15 Provider.TResolverErrorEvent Type

Represents the procedure type for the events that occur when the provider fails to update a record.

Pascal

```
TResolverErrorEvent = procedure (Sender: TObject; DataSet: TCustomClientDataSet; E:
EUpdateError; UpdateKind: TUpdateKind; var Response: TResolverResponse) of object;
```

C++

```
(Sender: TObject; DataSet: TCustomClientDataSet; E: EUpdateError; UpdateKind: TUpdateKind; var
Response: TResolverResponse) ( TResolverErrorEvent)();
```

File

Provider

Description

A variable of TResolverErrorEvent type references an event that occurs when the provider cannot update a record because of an error condition.

See Also

OnUpdateError (see page 3092)

Procedural types

Creating events

58.6 Variables

The following table lists variables in this documentation.

Variables

Name	Description
InformixLob (see page 3231)	Resolves Informix rather than Oracle BLOB and CLOB fields.

58.6.1 Provider.InformixLob Variable

Resolves Informix rather than Oracle BLOB and CLOB fields.

Pascal

```
InformixLob: Boolean;
```

C++

```
Boolean InformixLob;
```

File

Provider

Description

Informix BLOB (fldBLOB, fldstHBINARY) and CLOB (fldBLOB, fldstHMEMO) fields are mapped as ftOraBlob and ftOraClob respectively. The DataSnap resolver performs Oracle-specific query generation when resolving Oracle BLOB/CLOB fields, but not when resolving Informix BLOB/CLOBs. When using Informix BLOB/CLOBs, set InformixLob to true. To resume using Oracle BLOB/CLOBs, set InformixLob to false (the default).

58.7 Constants

The following table lists constants in this documentation.

Constants

Name	Description
MetaDataOption (see page 3232)	Specifies the options available when retrieving metadata from a target database.
ResetOption (see page 3233)	This is constant Provider.ResetOption.
XMLOption (see page 3233)	This is constant Provider.XMLOption.
XMLUTF8Option (see page 3234)	This is constant Provider.XMLUTF8Option.

58.7.1 Provider.MetaDataOption Constant

Specifies the options available when retrieving metadata from a target database.

Pascal

```
MetaDataOption: Integer = 1 shl ord(grMetaData);
```

C++

```
int MetaDataOption = 1 shl ord(grMetaData);
```

File

Provider

Description

Each database provides a number of options for data that can be retrieved from and about the database. When you build your own data provider, it can be helpful to use these options when implementing the `ISQLMetaData` interface. The following table describes the allowable metadata options.

Element	Description
CatalogName	Specifies a system table name.
DatabaseName	Specifies the name of the database.
DatabaseVersion	Specifies the version number of the database.
MaxColumnInSelect	Limits the number of columns that can be specified in a SELECT statement. The column count starts with the first column. Any columns beyond the limit are not returned.
MaxColumnsInTable	Limits the number of columns that can be returned from a table. The column count starts with the first column. Any columns beyond the limit are not returned.
MaxObjectNameLength	Limits the number of characters returned of a database object name. Names exceeding this limit are truncated. If database object names are similar, truncating them might result in duplicate names or meaningless names.
MaxRowSize	Limits the size of any returned rows. This can be useful if you suspect there might be BLOBs in the rows you are selecting, but you want to protect the performance of your query.
MaxSQLLength	Limits the length, in characters, of a SQL statement.
ObjectQuoteChar	Specifies the quote character to use when delimiting object names containing special characters, such as spaces.

PackageName	Specifies the name of a package.
ProcSupportsCursor	Specifies whether or not a stored procedure supports the use of a cursor. Set to **true** if it supports a cursor, otherwise set to **false**.
ProcSupportsCursors	Specifies whether or not a stored procedure supports the use of multiple cursors. Set to **true** if it supports cursors, otherwise set to **false**.
SQLEscapeChar	Specifies the character to use to delimit escape sequences in SQL statements.
SchemaName	Specifies the name of a database schema if different from the DatabaseName.
SupportsTransaction	Specifies whether or not the database can be named in a transaction-based SQL statement. Set to **true** if it supports a transaction, otherwise set to **false**.
SupportsTransactions	Specifies whether or not the database can be named in multiple transaction-based SQL statements. Set to **true** if it supports transactions, otherwise set to **false**.
TransactionIsoLevel	Specifies the transaction isolation level. Isolation levels supported by BDP.NET include ReadCommitted, RepeatableRead, DirtyRead, and Custom.

58.7.2 Provider.ResetOption Constant

Pascal

```
ResetOption: Integer = 1 shl ord(grReset);
```

C++

```
int ResetOption = 1 shl ord(grReset);
```

File

Provider

Description

This is constant Provider.ResetOption.

58.7.3 Provider.XMLOption Constant

Pascal

```
XMLOption: Integer = 1 shl ord(grXML);
```

C++

```
int XMLOption = 1 shl ord(grXML);
```

File

Provider

Description

This is constant Provider.XMLOption.

58.7.4 Provider.XMLUTF8Option Constant

Pascal

```
XMLUTF8Option: Integer = 1 shl ord(grXMLUTF8);
```

C++

```
int XMLUTF8Option = 1 shl ord(grXMLUTF8);
```

File

Provider

Description

This is constant Provider.XMLUTF8Option.

59 Registry Namespace

This is namespace Registry.

Topics

Name	Description
Classes (see page 3236)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 3327)	The following table lists structs, records, enums in this documentation.

Classes

	Name	Description
	ERegistryException (see page 3236)	ERegistryException is the exception class for registry errors.
	TRegIniFile (see page 3237)	TRegIniFile is a low-level wrapper for the Windows 95/NT system registry.
	TRegistry (see page 3246)	TRegistry is a low-level wrapper for the system registry and functions that operate on the registry.
	TRegistryIniFile (see page 3299)	TRegistryIniFile is a wrapper for the Windows system registry.

Structs, Records, Enums

	Name	Description
	TRegDataInfo (see page 3327)	TRegDataInfo is a type used by the GetDataInfo method of TRegistery.
	TRegDataType (see page 3327)	TRegDataType is a type used by the GetDataInfo method of TRegistery.
	TRegKeyInfo (see page 3328)	TRegKeyInfo is a type used by the GetRegKeyInfo method of TRegistery.

59.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	ERegistryException (see page 3236)	ERegistryException is the exception class for registry errors.
	TRegIniFile (see page 3237)	TRegIniFile is a low-level wrapper for the Windows 95/NT system registry.
	TRegistry (see page 3246)	TRegistry is a low-level wrapper for the system registry and functions that operate on the registry.
	TRegistryIniFile (see page 3299)	TRegistryIniFile is a wrapper for the Windows system registry.

59.1.1 ERegistryException Class

ERegistryException is the exception class for registry errors.

Class Hierarchy

Pascal

```
ERegistryException = class(Exception);
```

C++

```
class ERegistryException : public Exception;
```

File

Registry

Description

ERegistryException is raised when an application cannot

Create a new key in the registry.

Read a data value associated with a key.

Read a binary data value associated with a key into a local buffer.

Write a data value to a key.

Key creation fails when an application attempts to create a subkey under a key for which it does not have KEY_CREATE_SUB_KEY security access.

Reading a data value associated with a key fails when an application does not have KEY_READ security access, or when the application attempts to read the wrong kind of data from a key (for example, attempting to read string data from an integer data value).

Reading a binary data value into a local buffer fails when an application does not allocate a buffer large enough for the data.

Writing a data value to a key fails because an application does not have KEY_WRITE security access for the current key.

See Also

TRegistry (see page 3246)

TRegIniFile (see page 3237)

59.1.2 TRegIniFile Class

TRegIniFile is a low-level wrapper for the Windows 95/NT system registry.

Class Hierarchy

Pascal

```
TRegIniFile = class(TRegistry);
```

C++

```
class TRegIniFile : public TRegistry;
```

File

Registry

Description

TRegIniFile presents a simple interface to the system registry, hiding the need to know about the underlying structure of the registry. TRegIniFile acts as a helper object to TRegistryIniFile (see page 3299), which descends from TCustomIniFile (see page 716) (and is therefore more widely compatible with other INI file classes).

TRegIniFile enables handling the Windows 95/NT system registry as if it were a Windows 3.x INI file. Instead of processing an INI file, however, TRegIniFile reads from and writes to the system registry.

Although TRegIniFile descends from TRegistry (see page 3246), it adds the properties and methods used by a TIniFile (see page 752) object. The FileName (see page 3246) passed to a TRegIniFile object becomes a subkey under the system registry's root key (HKEY_CURRENT_USER by default). What corresponds to a section in an INI file is treated as a key in the system registry, and what corresponds to data entries under a section in an INI file are treated as data values under a key in the system registry.

Note: TRegIniFile is intended to be used primarily as a helper object to TRegistryIniFile (see page 3299), which unifies the interface to INI files and the system registry. However, applications can use TRegIniFile directly.

See Also

TIniFile (see page 752)

TMemIniFile (see page 765)

TCustomIniFile (see page 716)

TRegistryIniFile (see page 3299)

59.1.2.1 TRegIniFile Members

The following tables list the members exposed by TRegIniFile.

Protected Methods

	Name	Description
	ChangeKey (see page 3255)	This is ChangeKey, a member of class TRegistry.
	GetBaseKey (see page 3262)	This is GetBaseKey, a member of class TRegistry.

	GetData (see page 3262)	This is GetData, a member of class TRegistry.
	GetKey (see page 3264)	This is GetKey, a member of class TRegistry.
	PutData (see page 3273)	This is PutData, a member of class TRegistry.
	SetCurrentKey (see page 3280)	This is SetCurrentKey, a member of class TRegistry.

Public Methods

	Name	Description
	CloseKey (see page 3255)	Writes the current key to the registry and closes the key.
	Create (see page 3239)	This is the overview for the Create constructor overload.
	Create (see page 3251)	This is the overview for the Create constructor overload.
	CreateKey (see page 3258)	Creates a new key in the registry.
	DeleteKey (see page 3240)	Erases a data value that is associated with a key.
	DeleteValue (see page 3261)	Erases a data value that is associated with the current key.
	Destroy (see page 3253)	Closes the current key and destroys the TRegistry (see page 3246) object.
	EraseSection (see page 3241)	Deletes a key and its data values in the system registry.
	GetDataAsString (see page 3262)	This is GetDataAsString, a member of class TRegistry.
	GetDataInfo (see page 3263)	Returns information about a specified data value associated with the current key.
	GetDataSize (see page 3263)	Returns the size, in bytes, of a specified data value associated with the current key.
	GetDataType (see page 3264)	Returns the enumerated data type of a specified data value associated with the current key.
	GetKeyInfo (see page 3264)	Returns information about the current key.
	GetKeyNames (see page 3265)	Returns a string list containing the names of all subkeys belonging to the current key.
	GetValueNames (see page 3265)	Returns a string list containing the names of all data values associated with the current key.
	HasSubKeys (see page 3266)	Determines if the current key has subkeys.
	KeyExists (see page 3266)	Determines if a specified key exists.
	LoadKey (see page 3268)	Creates a subkey under the root key and loads registry information from a file into the newly created subkey.
	MoveKey (see page 3269)	Moves an existing key, its subkeys, and data values to a new location using a new key name.
	OpenKey (see page 3270)	Opens a specified key.
	OpenKeyReadOnly (see page 3272)	Opens a specified key in a read-only manner.
	ReadBinaryData (see page 3273)	Retrieves a binary value from a specified data value associated with the current key.
	ReadBool (see page 3241)	Retrieves a Boolean value from a specified data value associated with a key.
	ReadCurrency (see page 3274)	Retrieves a currency value from a specified data value associated with the current key.
	ReadDate (see page 3274)	Retrieves a date value from a specified data value associated with the current key.
	ReadDateTime (see page 3275)	Retrieves a date and time value from a specified data value associated with the current key.
	ReadFloat (see page 3275)	Retrieves a float value from a specified data value associated with the current key.
	ReadInteger (see page 3242)	Retrieves an integer value from a specified data value (Ident) associated with a key (Section).
	ReadSection (see page 3242)	Retrieves the names of all data values associated with a key, and stores the names in a TStrings object.
	ReadSections (see page 3243)	Retrieves the names of all subkeys associated with the current key.
	ReadSectionValues (see page 3243)	Retrieves the data-values associated with a specified key, and stores the retrieved values in a string object.
	ReadString (see page 3244)	Retrieves a string value from a specified data value (Ident) associated with a key (Section).
	ReadTime (see page 3277)	Retrieves a time value from a specified data value associated with the current key.
	RegistryConnect (see page 3277)	Establishes a connection to a registry on another computer.
	RenameValue (see page 3278)	Changes the name of a data value associated with the current key.

	ReplaceKey (see page 3278)	Replaces the hive file backing a key and all its subkeys with a hive stored in a different file, so that when the system is next started, the key and subkeys will have the values stored in the new file.
	RestoreKey (see page 3279)	Reopens the specified key, and overwrites any information contained in this key and its subkeys keys with registry data stored in a hive file.
	SaveKey (see page 3279)	Opens the specified key with the security access value KEY_ALL_ACCESS and saves the specified key and all of its subkeys and values to a hive file.
	UnLoadKey (see page 3280)	Removes a hive from the registry.
	ValueExists (see page 3281)	Determines if a specified data value exists for the current key.
	WriteBinaryData (see page 3281)	Stores a specifically named data value associated with the current key.
	WriteBool (see page 3244)	Writes a Boolean value to a specified data value associated with a specified key.
	WriteCurrency (see page 3282)	Stores a currency value in a specifically named data value associated with the current key.
	WriteDate (see page 3282)	Stores a date value in a specifically named data value associated with the current key.
	WriteDateTime (see page 3283)	Stores a datetime value in a specifically named data value associated with the current key.
	WriteExpandString (see page 3286)	Stores a string containing unexpanded references to environment variables in a specifically named data value associated with the current key.
	WriteFloat (see page 3288)	Stores a Double value in a specifically named data value associated with the current key.
	WriteInteger (see page 3245)	Writes a long integer value to a specified data value associated with a specified key.
	WriteString (see page 3245)	Writes a string value to a specified data value associated with a specified key.
	WriteTime (see page 3292)	Stores a time value in a specifically named data value associated with the current key.

Public Properties

	Name	Description
	Access (see page 3293)	Specifies the level of access to use when opening keys.
R	CurrentKey (see page 3296)	Specifies the registry key that is currently open.
R	CurrentPath (see page 3296)	Indicates the registry path associated with the current key.
R	FileName (see page 3246)	Specifies the name of the key opened or created in the system registry.
	LazyWrite (see page 3296)	Specifies how keys are written to the registry when calling the CloseKey (see page 3255) procedure.
	RootKey (see page 3297)	Specifies the root key for the TRegistry (see page 3246) object in an application.

59.1.2.2 Create Constructor

This is the overview for the Create constructor overload.

Overload List

	Name	Description
	TRegIniFile.Create (string) (see page 3239)	Creates a TRegIniFile (see page 3237) object for an application.
	TRegIniFile.Create (string, LongWord) (see page 3239)	Creates a TRegIniFile (see page 3237) object for an application.

59.1.2.2.1 TRegIniFile.Create Constructor (string)

Creates a TRegIniFile (see page 3237) object for an application.

Pascal

```
constructor Create(const FileName: string); overload;
```

```
constructor Create(const FileName: string; AAccess: LongWord); overload;
```

C++

```
__fastcall TRegIniFile(const UnicodeString FileName);
__fastcall TRegIniFile(const UnicodeString FileName, LongWord AAccess);
```

Description

Use Create to construct a TRegIniFile (see page 3237) object for an application.

FileName (see page 3246) initializes the FileName (see page 3246) property.

AAccess initializes the Access (see page 3293) property, which determines the access rights used when opening keys. If no AAccess parameter is supplied, Access (see page 3293) is initialized to KEY_ALL_ACCESS.

See Also

FileName (see page 3246)

Create (see page 3251)

Access (see page 3293)

59.1.2.3 TRegIniFile Methods

The methods of the TRegIniFile class are listed here.

Public Methods

	Name	Description
	DeleteKey (see page 3240)	Erases a data value that is associated with a key.
	EraseSection (see page 3241)	Deletes a key and its data values in the system registry.
	ReadBool (see page 3241)	Retrieves a Boolean value from a specified data value associated with a key.
	ReadInteger (see page 3242)	Retrieves an integer value from a specified data value (Ident) associated with a key (Section).
	ReadSection (see page 3242)	Retrieves the names of all data values associated with a key, and stores the names in a TStrings object.
	ReadSections (see page 3243)	Retrieves the names of all subkeys associated with the current key.
	ReadSectionValues (see page 3243)	Retrieves the data-values associated with a specified key, and stores the retrieved values in a string object.
	ReadString (see page 3244)	Retrieves a string value from a specified data value (Ident) associated with a key (Section).
	WriteBool (see page 3244)	Writes a Boolean value to a specified data value associated with a specified key.
	WriteInteger (see page 3245)	Writes a long integer value to a specified data value associated with a specified key.
	WriteString (see page 3245)	Writes a string value to a specified data value associated with a specified key.

59.1.2.3.1 TRegIniFile.DeleteKey Method

Erases a data value that is associated with a key.

Pascal

```
procedure DeleteKey(const Section: string; const Ident: string);
```

C++

```
__fastcall DeleteKey(const UnicodeString Section, const UnicodeString Ident);
```

Description

Call DeleteKey to erase a data value associated with a key. Section is string containing the name of the system registry key containing the value to delete. Ident is a string containing the name of the data value to delete.

Note: DeleteKey for TRegIniFile (see page 3237) obscures the DeleteKey method of TRegistry (see page 3246), from which TRegIniFile (see page 3237) descends. You can access TRegistry (see page 3246)'s DeleteKey method by casting the call to DeleteKey as type TRegistry (see page 3246), or by using the scope resolution operator.

Note: On Windows 95, DeleteKey will delete the key and all its subkeys. On Windows NT, the subkeys must be explicitly deleted by separate calls to DeleteKey.

See Also

DeleteKey (see page 3259)

EraseSection (see page 3241)

59.1.2.3.2 TRegIniFile.EraseSection Method

Deletes a key and its data values in the system registry.

Pascal

```
procedure EraseSection(const Section: string);
```

C++

```
__fastcall EraseSection(const UnicodeString Section);
```

Description

Call EraseSection to remove a key and its data values from the system registry. Section identifies the key to delete.

See Also

DeleteKey (see page 3240)

59.1.2.3.3 TRegIniFile.ReadBool Method

Retrieves a Boolean value from a specified data value associated with a key.

Pascal

```
function ReadBool(const Section: string; const Ident: string; Default: Boolean): Boolean;
```

C++

```
__fastcall Boolean ReadBool(const UnicodeString Section, const UnicodeString Ident, Boolean
Default);
```

Description

Call ReadBool to retrieve a Boolean value from a specified data value associated with a key.

Section is a string that identifies the key from which to retrieve a data value. Ident is a string that identifies the name of the data value to return. Default is a Boolean value to use if there is no key corresponding to Section or no data value corresponding to Ident.

Note: This function overrides the ReadBool function of TRegistry (see page 3246), from which TRegIniFile (see page 3237) is derived. To call the ReadBool function of TRegistry (see page 3246) instead of this function, cast the call to ReadBool as TRegistry (see page 3246).

See Also

ReadInteger (see page 3242)

ReadString (see page 3244)

ReadBool (see page 3274)

59.1.2.3.4 TRegIniFile.ReadInteger Method

Retrieves an integer value from a specified data value (Ident) associated with a key (Section).

Pascal

```
function ReadInteger(const Section: string; const Ident: string; Default: Longint): Longint;
```

C++

```
__fastcall Longint ReadInteger(const UnicodeString Section, const UnicodeString Ident, Longint
Default);
```

Description

Call ReadInteger to retrieve an integer value from a specified data value associated with a key.

Section is a string that identifies the key from which to retrieve a data value. Ident is a string that identifies the name of the data value to return. Default is an integer value to use if there is no key corresponding to Section or no data value corresponding to Ident.

Note: This function overrides the ReadInteger function of TRegistry (see page 3246), from which TRegIniFile (see page 3237) is derived. To call the ReadInteger function of TRegistry (see page 3246) instead of this function, cast the call to ReadInteger as TRegistry (see page 3246).

See Also

ReadBool (see page 3241)

ReadString (see page 3244)

ReadInteger (see page 3276)

59.1.2.3.5 TRegIniFile.ReadSection Method

Retrieves the names of all data values associated with a key, and stores the names in a TStrings object.

Pascal

```
procedure ReadSection(const Section: string; Strings: TStrings);
```

C++

```
__fastcall ReadSection(const UnicodeString Section, TStrings Strings);
```

Description

Call ReadSection to retrieve the names of all data values associated with a key into a TStrings object.

Section is a string identifying the key for which to retrieve data-value names.

Strings specifies the string object to hold the retrieved names. Strings can point to a TStrings object such as a string list, or to a component property, such as Items for a TListBox component, that is a TStrings object itself.

See Also

ReadSections (see page 3243)

ReadSectionValues (see page 3243)

TListBox

Items

59.1.2.3.6 TRegIniFile.ReadSections Method

Retrieves the names of all subkeys associated with the current key.

Pascal

```
procedure ReadSections(Strings: TStrings);
```

C++

```
__fastcall ReadSections(TStrings Strings);
```

Description

Call ReadSections to retrieve the names of all subkeys associated with the current key.

Strings specifies the string object to hold the retrieved names. Strings can point to a TStrings object, or to a component property, such as Items for a TListBox component, that is a TStrings object itself.

See Also

CurrentKey (see page 3296)

ReadSection (see page 3242)

ReadSectionValues (see page 3243)

TListBox

Items

59.1.2.3.7 TRegIniFile.ReadSectionValues Method

Retrieves the data-values associated with a specified key, and stores the retrieved values in a string object.

Pascal

```
procedure ReadSectionValues(const Section: string; Strings: TStrings);
```

C++

```
__fastcall ReadSectionValues(const UnicodeString Section, TStrings Strings);
```

Description

Call ReadSectionValues to read all data values associated with a specified key into a string object.

Section is the name of the key from which to retrieve data-values.

Strings specifies the string list object to hold the retrieved values. Strings can point to a TStrings object, or to a component property, such as Items for a TListBox component, that is a TStrings object itself. Use the Values property of the string object to access a specific string in the object.

See Also

ReadSection (see page 3242)

ReadSections (see page 3243)

TListBox

Items

Values

59.1.2.3.8 TRegIniFile.ReadString Method

Retrieves a string value from a specified data value (Ident) associated with a key (Section).

Pascal

```
function ReadString(const Section: string; const Ident: string; const Default: string): string;
```

C++

```
__fastcall UnicodeString ReadString(const UnicodeString Section, const UnicodeString Ident,
const UnicodeString Default);
```

Description

Call ReadString to retrieve a string value from a specified data value associated with a key.

Section is a string that identifies the key from which to retrieve a data value. Ident is a string that identifies the name of the data value to return. Default is a string value to use if there is no key corresponding to Section or no data value corresponding to Ident.

Note: This function overrides the ReadString function of TRegistry (see page 3246), from which TRegIniFile (see page 3237) is derived. To call the ReadString function of TRegistry (see page 3246) instead of this function, cast the call to ReadString as TRegistry (see page 3246).

See Also

ReadBool (see page 3241)

ReadInteger (see page 3242)

ReadString (see page 3276)

59.1.2.3.9 TRegIniFile.WriteBool Method

Writes a Boolean value to a specified data value associated with a specified key.

Pascal

```
procedure WriteBool(const Section: string; const Ident: string; Value: Boolean);
```

C++

```
__fastcall WriteBool(const UnicodeString Section, const UnicodeString Ident, Boolean Value);
```

Description

Call WriteBool to store a Boolean value in a data value associated with a specified key.

Section is a string identifying the key into which to store a data value. Ident is a string identifying the name of the data value into which to write. Value is the Boolean value to write into the data value.

Note: This function overrides the WriteBool procedure of TRegistry (see page 3246), from which TRegIniFile (see page 3237) is derived. To call the WriteBool procedure of TRegistry (see page 3246) instead of this procedure, cast the call to WriteBool as TRegistry (see page 3246).

See Also

WriteInteger (see page 3245)

WriteString (see page 3245)

WriteBool (see page 3282)

59.1.2.3.10 TRegIniFile.WriteInteger Method

Writes a long integer value to a specified data value associated with a specified key.

Pascal

```
procedure WriteInteger(const Section: string; const Ident: string; Value: Longint);
```

C++

```
__fastcall WriteInteger(const UnicodeString Section, const UnicodeString Ident, Longint Value);
```

Description

Call WriteInteger to store an integer value in a data value associated with a specified key.

Section is a string identifying the key into which to store a data value. Ident is a string identifying the name of the data value into which to write. Value is the integer value to write into the data value.

Note: This function overrides the WriteInteger procedure of TRegistry (see page 3246), from which TRegIniFile (see page 3237) is derived. To call the WriteInteger procedure of TRegistry (see page 3246) instead of this procedure, cast the call to WriteInteger as TRegistry (see page 3246).

See Also

WriteBool (see page 3244)

WriteString (see page 3245)

WriteInteger (see page 3289)

59.1.2.3.11 TRegIniFile.WriteString Method

Writes a string value to a specified data value associated with a specified key.

Pascal

```
procedure WriteString(const Section: string; const Ident: string; const Value: string);
```

C++

```
__fastcall WriteString(const UnicodeString Section, const UnicodeString Ident, const
UnicodeString Value);
```

Description

Call WriteString to store a string value in a data value associated with a specified key.

Section is a string identifying the key into which to store a data value. Ident is a string identifying the name of the data value into which to write. Value is the string value to write into the data value.

Note: This function overrides the WriteString procedure of TRegistry (see page 3246), from which TRegIniFile (see page 3237) is derived. To call the WriteString procedure of TRegistry (see page 3246) instead of this procedure, cast the call to WriteString as TRegistry (see page 3246).

See Also

WriteBool (see page 3244)

WriteInteger (see page 3245)

WriteString (see page 3289)

59.1.2.4 TRegIniFile Properties

The properties of the TRegIniFile class are listed here.

Public Properties

	Name	Description
	FileName (see page 3246)	Specifies the name of the key opened or created in the system registry.

59.1.2.4.1 TRegIniFile.FileName Property

Specifies the name of the key opened or created in the system registry.

Pascal

```
property FileName: string;
```

C++

```
__property UnicodeString FileName;
```

Description

Query FileName to determine the name of the key in the system registry that is the root key for all TRegIniFile (see page 3237) operations.

See Also

CurrentKey (see page 3296)

RootKey (see page 3297)

59.1.3 TRegistry Class

TRegistry is a low-level wrapper for the system registry and functions that operate on the registry.

Class Hierarchy

Pascal

```
TRegistry = class(TObject);
```

C++

```
class TRegistry : public TObject;
```

File

Registry

Description

Use TRegistry to encapsulate access to the Windows system registry in an application. The registry is a database that an application can use to store and retrieve configuration information. Configuration information is stored in a hierarchical tree. Each node in the tree is called a key. Every key can contain subkeys and data values that represent part of the configuration information for an application.

All keys that an application creates, opens, reads, or writes are subkeys of predefined root keys. By default, a TRegistry object is created with a root key of HKEY_CURRENT_USER.

Only one key is accessible at a time in a TRegistry object. To determine the key that is currently accessible, read the value of the CurrentKey (see page 3296) property. TRegistry methods enable an application to open, close, save, move, copy, and delete keys.

One or more data values containing actual configuration information can be stored in a key. TRegistry methods enable an application to query a key to see if it contains data, to read data in a key, and to write data to a key.

Note: The TRegistry component is not fully compatable with the Windows NT environment. Functions such as RestoreKey (see page 3279) and SaveKey (see page 3279) may not perform properly without a significant number of workarounds.

C++ Examples:

```
/*
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
for inserting and deleting the values.
*/
void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
{
    TRegistry* reg = new TRegistry(KEY_READ);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
    {
        MessageDlg("Key not found! Created now.",
                   mtInformation, TMsgDlgButtons() << mbOK, 0);
    }
    reg->Access = KEY_WRITE;
    bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);

    if(!openResult)
    {
        MessageDlg("Unable to create key! Exiting.",
                   mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }
    //checking if the values exist and inserting when neccesary

    if(!reg->KeyExists("Creation\ Date"))
    {
        TDateTime today = TDateTime::CurrentDateTime();
        reg->WriteDateTime("Creation\ Date", today);
    }

    if(!reg->KeyExists("Licenced\ To"))
    {
        reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
    }

    if(!reg->KeyExists("App\ Location"))
    {
        reg->WriteExpandString("App\ Location",
                               "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
    }

    if(!reg->KeyExists("Projects\ Location"))
    {
        reg->WriteExpandString("Projects\ Location",
                               "%USERPROFILE%\\MyApplication\\Projects\\");
    }

    reg->CloseKey();
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
{
    //deleting the example registry entries
    TRegistry* reg = new TRegistry(KEY_WRITE);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
    reg->DeleteKey("Software\\MyCompanyName");

    reg->CloseKey();
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
on a form, for inserting and deleting the values.
}
procedure TForm3.InsertToRegBtnClick(Sender: TObject);
var
  reg        : TRegistry;
  openResult : Boolean;
  today      : TDateTime;
begin
  reg := TRegistry.Create(KEY_READ);
  reg.RootKey := HKEY_LOCAL_MACHINE;

  if (not reg.KeyExists('Software\\MyCompanyName\\MyApplication\\')) then
    begin
      MessageDlg('Key not found! Created now.',
                           mtInformation, mbOKCancel, 0);
    end;
  reg.Access := KEY_WRITE;
  openResult := reg.OpenKey('Software\\MyCompanyName\\MyApplication\\',True);

  if not openResult = True then
    begin
      MessageDlg('Unable to create key! Exiting.',
                 mtError, mbOKCancel, 0);
      Exit();
    end;

  {checking if the values exist and inserting when neccesary}

  if not reg.KeyExists('Creation\ Date') then
    begin
      today := Now;
        reg.WriteDateTime('Creation\ Date', today);
    end;

  if not reg.KeyExists('Licenced\ To') then
    begin
        reg.WriteString('Licenced\ To', 'MySurname\ MyFirstName');
    end;

  if not reg.KeyExists('App\ Location') then
    begin
        reg.WriteExpandString('App\ Location',
                              '%PROGRAMFILES%\\MyCompanyName\\MyApplication\\');
    end;

  if not reg.KeyExists('Projects\ Location') then
```

```
    begin
        reg.WriteExpandString('Projects\ Location',
                              '%USERPROFILE%\\MyApplication\\Projects\\');
    end;

  reg.CloseKey();

end;

procedure TForm3.DeleteFromRegBtnClick(Sender: TObject);
var
  reg : TRegistry;
begin
  reg := TRegistry.Create(KEY_WRITE);
  reg.RootKey := HKEY_LOCAL_MACHINE;

  reg.DeleteKey('Software\\MyCompanyName\\MyApplication');
  reg.DeleteKey('Software\\MyCompanyName');

  reg.CloseKey();
end;
```

59.1.3.1 TRegistry Members

The following tables list the members exposed by TRegistry.

Protected Methods

	Name	Description
	ChangeKey (see page 3255)	This is ChangeKey, a member of class TRegistry.
	GetBaseKey (see page 3262)	This is GetBaseKey, a member of class TRegistry.
	GetData (see page 3262)	This is GetData, a member of class TRegistry.
	GetKey (see page 3264)	This is GetKey, a member of class TRegistry.
	PutData (see page 3273)	This is PutData, a member of class TRegistry.
	SetCurrentKey (see page 3280)	This is SetCurrentKey, a member of class TRegistry.

Public Methods

	Name	Description
	CloseKey (see page 3255)	Writes the current key to the registry and closes the key.
	Create (see page 3251)	This is the overview for the Create constructor overload.
	CreateKey (see page 3258)	Creates a new key in the registry.
	DeleteKey (see page 3259)	Removes a specified key and its associated data from the registry.
	DeleteValue (see page 3261)	Erases a data value that is associated with the current key.
	Destroy (see page 3253)	Closes the current key and destroys the TRegistry (see page 3246) object.
	GetDataAsString (see page 3262)	This is GetDataAsString, a member of class TRegistry.
	GetDataInfo (see page 3263)	Returns information about a specified data value associated with the current key.
	GetDataSize (see page 3263)	Returns the size, in bytes, of a specified data value associated with the current key.
	GetDataType (see page 3264)	Returns the enumerated data type of a specified data value associated with the current key.
	GetKeyInfo (see page 3264)	Returns information about the current key.
	GetKeyNames (see page 3265)	Returns a string list containing the names of all subkeys belonging to the current key.
	GetValueNames (see page 3265)	Returns a string list containing the names of all data values associated with the current key.
	HasSubKeys (see page 3266)	Determines if the current key has subkeys.
	KeyExists (see page 3266)	Determines if a specified key exists.
	LoadKey (see page 3268)	Creates a subkey under the root key and loads registry information from a file into the newly created subkey.

	MoveKey (see page 3269)	Moves an existing key, its subkeys, and data values to a new location using a new key name.
	OpenKey (see page 3270)	Opens a specified key.
	OpenKeyReadOnly (see page 3272)	Opens a specified key in a read-only manner.
	ReadBinaryData (see page 3273)	Retrieves a binary value from a specified data value associated with the current key.
	ReadBool (see page 3274)	Retrieves a Boolean value from a specified data value associated with the current key.
	ReadCurrency (see page 3274)	Retrieves a currency value from a specified data value associated with the current key.
	ReadDate (see page 3274)	Retrieves a date value from a specified data value associated with the current key.
	ReadDateTime (see page 3275)	Retrieves a date and time value from a specified data value associated with the current key.
	ReadFloat (see page 3275)	Retrieves a float value from a specified data value associated with the current key.
	ReadInteger (see page 3276)	Retrieves an integer value from a specified data value associated with the current key.
	ReadString (see page 3276)	Retrieves a string value from a specified data value associated with the current key.
	ReadTime (see page 3277)	Retrieves a time value from a specified data value associated with the current key.
	RegistryConnect (see page 3277)	Establishes a connection to a registry on another computer.
	RenameValue (see page 3278)	Changes the name of a data value associated with the current key.
	ReplaceKey (see page 3278)	Replaces the hive file backing a key and all its subkeys with a hive stored in a different file, so that when the system is next started, the key and subkeys will have the values stored in the new file.
	RestoreKey (see page 3279)	Reopens the specified key, and overwrites any information contained in this key and its subkeys keys with registry data stored in a hive file.
	SaveKey (see page 3279)	Opens the specified key with the security access value KEY_ALL_ACCESS and saves the specified key and all of its subkeys and values to a hive file.
	UnLoadKey (see page 3280)	Removes a hive from the registry.
	ValueExists (see page 3281)	Determines if a specified data value exists for the current key.
	WriteBinaryData (see page 3281)	Stores a specifically named data value associated with the current key.
	WriteBool (see page 3282)	Stores a Boolean value in a specifically named data value associated with the current key.
	WriteCurrency (see page 3282)	Stores a currency value in a specifically named data value associated with the current key.
	WriteDate (see page 3282)	Stores a date value in a specifically named data value associated with the current key.
	WriteDateTime (see page 3283)	Stores a datetime value in a specifically named data value associated with the current key.
	WriteExpandString (see page 3286)	Stores a string containing unexpanded references to environment variables in a specifically named data value associated with the current key.
	WriteFloat (see page 3288)	Stores a Double value in a specifically named data value associated with the current key.
	WriteInteger (see page 3289)	Stores an integer value in a specifically named data value associated with the current key.
	WriteString (see page 3289)	Stores a string value in a specifically named data value associated with the current key.
	WriteTime (see page 3292)	Stores a time value in a specifically named data value associated with the current key.

Public Properties

	Name	Description
	Access (see page 3293)	Specifies the level of access to use when opening keys.
R	CurrentKey (see page 3296)	Specifies the registry key that is currently open.
R	CurrentPath (see page 3296)	Indicates the registry path associated with the current key.
	LazyWrite (see page 3296)	Specifies how keys are written to the registry when calling the CloseKey (see page 3255) procedure.
	RootKey (see page 3297)	Specifies the root key for the TRegistry (see page 3246) object in an application.

59.1.3.2 Create Constructor

This is the overview for the Create constructor overload.

Overload List

	Name	Description
	TRegistry.Create () (see page 3251)	Creates a TRegistry (see page 3246) object for an application.
	TRegistry.Create (LongWord) (see page 3251)	Creates a TRegistry (see page 3246) object for an application.

59.1.3.2.1 TRegistry.Create Constructor ()

Creates a TRegistry (see page 3246) object for an application.

Pascal

```
constructor Create; overload;
constructor Create(AAccess: LongWord); overload;
```

C++

```
__fastcall TRegistry();
__fastcall TRegistry(LongWord AAccess);
```

Description

Create instantiates a TRegistry (see page 3246) object and sets the RootKey (see page 3297) property to HKEY_CURRENT_USER and the LazyWrite (see page 3296) property to true. If an AAccess parameter is supplied, this value is used to initialize the Access (see page 3293) property. Otherwise, Access (see page 3293) is initialized to KEY_ALL_ACCESS.

C++ Examples:

```
/*
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
for inserting and deleting the values.
*/
void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
{
    TRegistry* reg = new TRegistry(KEY_READ);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
    {
        MessageDlg("Key not found! Created now.",
                   mtInformation, TMsgDlgButtons() << mbOK, 0);
    }
    reg->Access = KEY_WRITE;
    bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);

    if(!openResult)
    {
        MessageDlg("Unable to create key! Exiting.",
                   mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }
    //checking if the values exist and inserting when neccesary
```

```
    if(!reg->KeyExists("Creation\ Date"))
    {
        TDateTime today = TDateTime::CurrentDateTime();
        reg->WriteDateTime("Creation\ Date", today);
    }

    if(!reg->KeyExists("Licenced\ To"))
    {
        reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
    }

    if(!reg->KeyExists("App\ Location"))
    {
        reg->WriteExpandString("App\ Location",
                               "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
    }

    if(!reg->KeyExists("Projects\ Location"))
    {
        reg->WriteExpandString("Projects\ Location",
                               "%USERPROFILE%\\MyApplication\\Projects\\");
    }

    reg->CloseKey();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
{
    //deleting the example registry entries
    TRegistry* reg = new TRegistry(KEY_WRITE);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
    reg->DeleteKey("Software\\MyCompanyName");

    reg->CloseKey();
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
on a form, for inserting and deleting the values.
}
procedure TForm3.InsertToRegBtnClick(Sender: TObject);
var
  reg        : TRegistry;
  openResult : Boolean;
  today      : TDateTime;
begin
  reg := TRegistry.Create(KEY_READ);
  reg.RootKey := HKEY_LOCAL_MACHINE;

  if (not reg.KeyExists('Software\\MyCompanyName\\MyApplication\\')) then
    begin
      MessageDlg('Key not found! Created now.',
                          mtInformation, mbOKCancel, 0);
    end;
  reg.Access := KEY_WRITE;
  openResult := reg.OpenKey('Software\\MyCompanyName\\MyApplication\\',True);

  if not openResult = True then
    begin
```

```
        MessageDlg('Unable to create key! Exiting.',
                   mtError, mbOKCancel, 0);
        Exit();
      end;

    {checking if the values exist and inserting when neccesary}

    if not reg.KeyExists('Creation\ Date') then
      begin
        today := Now;
          reg.WriteDateTime('Creation\ Date', today);
      end;

    if not reg.KeyExists('Licenced\ To') then
      begin
          reg.WriteString('Licenced\ To', 'MySurname\ MyFirstName');
      end;

    if not reg.KeyExists('App\ Location') then
      begin
          reg.WriteExpandString('App\ Location',
                                '%PROGRAMFILES%\\MyCompanyName\\MyApplication\\');
      end;

    if not reg.KeyExists('Projects\ Location') then
      begin
          reg.WriteExpandString('Projects\ Location',
                                '%USERPROFILE%\\MyApplication\\Projects\\');
      end;

    reg.CloseKey();

  end;

  procedure TForm3.DeleteFromRegBtnClick(Sender: TObject);
  var
    reg : TRegistry;
  begin
    reg := TRegistry.Create(KEY_WRITE);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    reg.DeleteKey('Software\\MyCompanyName\\MyApplication');
    reg.DeleteKey('Software\\MyCompanyName');

    reg.CloseKey();
  end;
```

See Also

LazyWrite (see page 3296)

RootKey (see page 3297)

Destroy (see page 3253)

Access (see page 3293)

59.1.3.3 TRegistry.Destroy Destructor

Closes the current key and destroys the TRegistry (see page 3246) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRegistry();
```

Description

Destroy closes the current key and frees the resources allocated for a registry object when it is no longer needed.

See Also

CurrentKey (see page 3296)

Create (see page 3251)

System::TObject::Free

59.1.3.4 TRegistry Methods

The methods of the TRegistry class are listed here.

Protected Methods

	Name	Description
	ChangeKey (see page 3255)	This is ChangeKey, a member of class TRegistry.
	GetBaseKey (see page 3262)	This is GetBaseKey, a member of class TRegistry.
	GetData (see page 3262)	This is GetData, a member of class TRegistry.
	GetKey (see page 3264)	This is GetKey, a member of class TRegistry.
	PutData (see page 3273)	This is PutData, a member of class TRegistry.
	SetCurrentKey (see page 3280)	This is SetCurrentKey, a member of class TRegistry.

Public Methods

	Name	Description
	CloseKey (see page 3255)	Writes the current key to the registry and closes the key.
	CreateKey (see page 3258)	Creates a new key in the registry.
	DeleteKey (see page 3259)	Removes a specified key and its associated data from the registry.
	DeleteValue (see page 3261)	Erases a data value that is associated with the current key.
	GetDataAsString (see page 3262)	This is GetDataAsString, a member of class TRegistry.
	GetDataInfo (see page 3263)	Returns information about a specified data value associated with the current key.
	GetDataSize (see page 3263)	Returns the size, in bytes, of a specified data value associated with the current key.
	GetDataType (see page 3264)	Returns the enumerated data type of a specified data value associated with the current key.
	GetKeyInfo (see page 3264)	Returns information about the current key.
	GetKeyNames (see page 3265)	Returns a string list containing the names of all subkeys belonging to the current key.
	GetValueNames (see page 3265)	Returns a string list containing the names of all data values associated with the current key.
	HasSubKeys (see page 3266)	Determines if the current key has subkeys.
	KeyExists (see page 3266)	Determines if a specified key exists.
	LoadKey (see page 3268)	Creates a subkey under the root key and loads registry information from a file into the newly created subkey.
	MoveKey (see page 3269)	Moves an existing key, its subkeys, and data values to a new location using a new key name.
	OpenKey (see page 3270)	Opens a specified key.
	OpenKeyReadOnly (see page 3272)	Opens a specified key in a read-only manner.
	ReadBinaryData (see page 3273)	Retrieves a binary value from a specified data value associated with the current key.
	ReadBool (see page 3274)	Retrieves a Boolean value from a specified data value associated with the current key.

	ReadCurrency (see page 3274)	Retrieves a currency value from a specified data value associated with the current key.
	ReadDate (see page 3274)	Retrieves a date value from a specified data value associated with the current key.
	ReadDateTime (see page 3275)	Retrieves a date and time value from a specified data value associated with the current key.
	ReadFloat (see page 3275)	Retrieves a float value from a specified data value associated with the current key.
	ReadInteger (see page 3276)	Retrieves an integer value from a specified data value associated with the current key.
	ReadString (see page 3276)	Retrieves a string value from a specified data value associated with the current key.
	ReadTime (see page 3277)	Retrieves a time value from a specified data value associated with the current key.
	RegistryConnect (see page 3277)	Establishes a connection to a registry on another computer.
	RenameValue (see page 3278)	Changes the name of a data value associated with the current key.
	ReplaceKey (see page 3278)	Replaces the hive file backing a key and all its subkeys with a hive stored in a different file, so that when the system is next started, the key and subkeys will have the values stored in the new file.
	RestoreKey (see page 3279)	Reopens the specified key, and overwrites any information contained in this key and its subkeys keys with registry data stored in a hive file.
	SaveKey (see page 3279)	Opens the specified key with the security access value KEY_ALL_ACCESS and saves the specified key and all of its subkeys and values to a hive file.
	UnLoadKey (see page 3280)	Removes a hive from the registry.
	ValueExists (see page 3281)	Determines if a specified data value exists for the current key.
	WriteBinaryData (see page 3281)	Stores a specifically named data value associated with the current key.
	WriteBool (see page 3282)	Stores a Boolean value in a specifically named data value associated with the current key.
	WriteCurrency (see page 3282)	Stores a currency value in a specifically named data value associated with the current key.
	WriteDate (see page 3282)	Stores a date value in a specifically named data value associated with the current key.
	WriteDateTime (see page 3283)	Stores a datetime value in a specifically named data value associated with the current key.
	WriteExpandString (see page 3286)	Stores a string containing unexpanded references to environment variables in a specifically named data value associated with the current key.
	WriteFloat (see page 3288)	Stores a Double value in a specifically named data value associated with the current key.
	WriteInteger (see page 3289)	Stores an integer value in a specifically named data value associated with the current key.
	WriteString (see page 3289)	Stores a string value in a specifically named data value associated with the current key.
	WriteTime (see page 3292)	Stores a time value in a specifically named data value associated with the current key.

59.1.3.4.1 TRegistry.ChangeKey Method

Pascal

```
procedure ChangeKey(Value: HKey; const Path: string);
```

C++

```
__fastcall ChangeKey(HKey Value, const UnicodeString Path);
```

Description

This is ChangeKey, a member of class TRegistry.

59.1.3.4.2 TRegistry.CloseKey Method

Writes the current key to the registry and closes the key.

Pascal

```
procedure CloseKey;
```

C++

```
__fastcall CloseKey();
```

Description

Call CloseKey to write the current key to the registry and close the key. An application should not keep keys open any longer than necessary. Many TRegistry (see page 3246) methods routinely call CloseKey after reading from or writing to the registry. Calling CloseKey when there is no current key has no effect.

Note: When an application closes a key, the key is written to the registry. The LazyWrite (see page 3296) property determines how the write takes place.

C++ Examples:

```
/*
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
for inserting and deleting the values.
*/
void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
{
    TRegistry* reg = new TRegistry(KEY_READ);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
    {
        MessageDlg("Key not found! Created now.",
                   mtInformation, TMsgDlgButtons() << mbOK, 0);
    }
    reg->Access = KEY_WRITE;
    bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);

    if(!openResult)
    {
        MessageDlg("Unable to create key! Exiting.",
                   mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }
    //checking if the values exist and inserting when neccesary

    if(!reg->KeyExists("Creation\ Date"))
    {
        TDateTime today = TDateTime::CurrentDateTime();
        reg->WriteDateTime("Creation\ Date", today);
    }

    if(!reg->KeyExists("Licenced\ To"))
    {
        reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
    }

    if(!reg->KeyExists("App\ Location"))
    {
        reg->WriteExpandString("App\ Location",
                               "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
    }

    if(!reg->KeyExists("Projects\ Location"))
    {
```

```
        reg->WriteExpandString("Projects\ Location",
                               "%USERPROFILE%\\MyApplication\\Projects\\");
    }

    reg->CloseKey();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
{
    //deleting the example registry entries
    TRegistry* reg = new TRegistry(KEY_WRITE);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
    reg->DeleteKey("Software\\MyCompanyName");

    reg->CloseKey();
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
on a form, for inserting and deleting the values.
}
procedure TForm3.InsertToRegBtnClick(Sender: TObject);
var
  reg        : TRegistry;
  openResult : Boolean;
  today      : TDateTime;
begin
  reg := TRegistry.Create(KEY_READ);
  reg.RootKey := HKEY_LOCAL_MACHINE;

  if (not reg.KeyExists('Software\\MyCompanyName\\MyApplication\\')) then
    begin
      MessageDlg('Key not found! Created now.',
                           mtInformation, mbOKCancel, 0);
    end;
  reg.Access := KEY_WRITE;
  openResult := reg.OpenKey('Software\\MyCompanyName\\MyApplication\\',True);

  if not openResult = True then
    begin
      MessageDlg('Unable to create key! Exiting.',
                 mtError, mbOKCancel, 0);
      Exit();
    end;

  {checking if the values exist and inserting when neccesary}

  if not reg.KeyExists('Creation\ Date') then
    begin
      today := Now;
        reg.WriteDateTime('Creation\ Date', today);
    end;

  if not reg.KeyExists('Licenced\ To') then
    begin
        reg.WriteString('Licenced\ To', 'MySurname\ MyFirstName');
    end;

  if not reg.KeyExists('App\ Location') then
    begin
```

```
          reg.WriteExpandString('App\ Location',
                                '%PROGRAMFILES%\\MyCompanyName\\MyApplication\\');
      end;

    if not reg.KeyExists('Projects\ Location') then
      begin
          reg.WriteExpandString('Projects\ Location',
                                '%USERPROFILE%\\MyApplication\\Projects\\');
      end;

    reg.CloseKey();

  end;

  procedure TForm3.DeleteFromRegBtnClick(Sender: TObject);
  var
    reg : TRegistry;
  begin
    reg := TRegistry.Create(KEY_WRITE);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    reg.DeleteKey('Software\\MyCompanyName\\MyApplication');
    reg.DeleteKey('Software\\MyCompanyName');

    reg.CloseKey();
  end;
```

See Also

CurrentKey (see page 3296)

LazyWrite (see page 3296)

59.1.3.4.3 TRegistry.CreateKey Method

Creates a new key in the registry.

Pascal

```
function CreateKey(const Key: string): Boolean;
```

C++

```
__fastcall Boolean CreateKey(const UnicodeString Key);
```

Description

Use CreateKey to add a new key to the registry. Key is the name of the key to create. Key can be an absolute or relative name. An absolute key begins with a backslash (\) and is a subkey of the root key. A relative key is a subkey of the current key.

CreateKey returns true if key creation is successful. On error, an exception is raised. Attempting to create a key that already exists has no effect.

CreateKey creates only non-volatile keys. If you need to create volatile keys on Windows NT, you must call the Windows API directly.

CreateKey creates keys with a security access of KEY_ALL_ACCESS. If you need to create keys with more restricted access, you must call the Windows API directly.

A key created by CreateKey has no values. To set key values, use the WriteCurrency (see page 3282), WriteBinaryData (see page 3281), WriteBool (see page 3282), WriteDate (see page 3282), WriteDateTime (see page 3283), WriteFloat (see page 3288), WriteInteger (see page 3289), WriteString (see page 3289), or WriteTime (see page 3292) procedures.

See Also

WriteCurrency (see page 3282)

WriteBinaryData (see page 3281)

WriteBool (see page 3282)

WriteDate (see page 3282)

WriteDateTime (see page 3283)

WriteFloat (see page 3288)

WriteInteger (see page 3289)

WriteString (see page 3289)

WriteTime (see page 3292)

Access (see page 3293)

59.1.3.4.4 TRegistry.DeleteKey Method

Removes a specified key and its associated data from the registry.

Pascal

```
function DeleteKey(const Key: string): Boolean;
```

C++

```
__fastcall Boolean DeleteKey(const UnicodeString Key);
```

Description

Call DeleteKey to remove a specified key and its associated data, if any, from the registry. Under Windows 95, if the key has subkeys, the subkeys and any associated data are also removed. Under Windows NT, subkeys must be explicitly deleted by separate calls to DeleteKey.

DeleteKey returns true if key deletion is successful. On error, DeleteKey returns false.

C++ Examples:

```
/*
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
for inserting and deleting the values.
*/
void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
{
    TRegistry* reg = new TRegistry(KEY_READ);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
    {
        MessageDlg("Key not found! Created now.",
                    mtInformation, TMsgDlgButtons() << mbOK, 0);
    }
    reg->Access = KEY_WRITE;
    bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);

    if(!openResult)
    {
        MessageDlg("Unable to create key! Exiting.",
                    mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }
    //checking if the values exist and inserting when neccesary
```

```
    if(!reg->KeyExists("Creation\ Date"))
    {
        TDateTime today = TDateTime::CurrentDateTime();
        reg->WriteDateTime("Creation\ Date", today);
    }

    if(!reg->KeyExists("Licenced\ To"))
    {
        reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
    }

    if(!reg->KeyExists("App\ Location"))
    {
        reg->WriteExpandString("App\ Location",
                               "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
    }

    if(!reg->KeyExists("Projects\ Location"))
    {
        reg->WriteExpandString("Projects\ Location",
                               "%USERPROFILE%\\MyApplication\\Projects\\");
    }

    reg->CloseKey();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
{
    //deleting the example registry entries
    TRegistry* reg = new TRegistry(KEY_WRITE);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
    reg->DeleteKey("Software\\MyCompanyName");

    reg->CloseKey();
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
on a form, for inserting and deleting the values.
}
procedure TForm3.InsertToRegBtnClick(Sender: TObject);
var
  reg        : TRegistry;
  openResult : Boolean;
  today      : TDateTime;
begin
  reg := TRegistry.Create(KEY_READ);
  reg.RootKey := HKEY_LOCAL_MACHINE;

  if (not reg.KeyExists('Software\\MyCompanyName\\MyApplication\\')) then
    begin
      MessageDlg('Key not found! Created now.',
                          mtInformation, mbOKCancel, 0);
    end;
  reg.Access := KEY_WRITE;
  openResult := reg.OpenKey('Software\\MyCompanyName\\MyApplication\\',True);

  if not openResult = True then
    begin
      MessageDlg('Unable to create key! Exiting.',
```

```
                    mtError, mbOKCancel, 0);
        Exit();
      end;

    {checking if the values exist and inserting when neccesary}

    if not reg.KeyExists('Creation\ Date') then
      begin
        today := Now;
          reg.WriteDateTime('Creation\ Date', today);
      end;

    if not reg.KeyExists('Licenced\ To') then
      begin
          reg.WriteString('Licenced\ To', 'MySurname\ MyFirstName');
      end;

    if not reg.KeyExists('App\ Location') then
      begin
          reg.WriteExpandString('App\ Location',
                                '%PROGRAMFILES%\\MyCompanyName\\MyApplication\\');
      end;

    if not reg.KeyExists('Projects\ Location') then
      begin
          reg.WriteExpandString('Projects\ Location',
                                '%USERPROFILE%\\MyApplication\\Projects\\');
      end;

    reg.CloseKey();

  end;

  procedure TForm3.DeleteFromRegBtnClick(Sender: TObject);
  var
    reg : TRegistry;
  begin
    reg := TRegistry.Create(KEY_WRITE);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    reg.DeleteKey('Software\\MyCompanyName\\MyApplication');
    reg.DeleteKey('Software\\MyCompanyName');

    reg.CloseKey();
  end;
```

See Also

CreateKey (see page 3258)

DeleteValue (see page 3261)

59.1.3.4.5 TRegistry.DeleteValue Method

Erases a data value that is associated with the current key.

Pascal

```
function DeleteValue(const Name: string): Boolean;
```

C++

```
__fastcall Boolean DeleteValue(const UnicodeString Name);
```

Description

Call DeleteValue to remove a specific data value associated with the current key. Name is string containing the name of the

value to delete. Keys can contain multiple data values, and every value associated with a key has a unique name.

DeleteValue returns true if the deletion is successful, false if the deletion fails.

See Also

DeleteKey (see page 3259)

WriteCurrency (see page 3282)

WriteBinaryData (see page 3281)

WriteBool (see page 3282)

WriteDate (see page 3282)

WriteDateTime (see page 3283)

WriteFloat (see page 3288)

WriteInteger (see page 3289)

WriteString (see page 3289)

WriteTime (see page 3292)

59.1.3.4.6 TRegistry.GetBaseKey Method

Pascal

```
function GetBaseKey(Relative: Boolean): HKey;
```

C++

```
__fastcall HKey GetBaseKey(Boolean Relative);
```

Description

This is GetBaseKey, a member of class TRegistry.

59.1.3.4.7 TRegistry.GetData Method

Pascal

```
function GetData(const Name: string; Buffer: Pointer; BufSize: Integer; var RegData:
TRegDataType): Integer;
```

C++

```
__fastcall int GetData(const UnicodeString Name, void * Buffer, int BufSize, TRegDataType
RegData);
```

Description

This is GetData, a member of class TRegistry.

59.1.3.4.8 TRegistry.GetDataAsString Method

Pascal

```
function GetDataAsString(const ValueName: string; PrefixType: Boolean = false): string;
```

C++

```
__fastcall UnicodeString GetDataAsString(const UnicodeString ValueName, Boolean PrefixType =
false);
```

Description

This is GetDataAsString, a member of class TRegistry.

59.1.3.4.9 TRegistry.GetDataInfo Method

Returns information about a specified data value associated with the current key.

Pascal

```
function GetDataInfo(const ValueName: string; var Value: TRegDataInfo): Boolean;
```

C++

```
__fastcall Boolean GetDataInfo(const UnicodeString ValueName, TRegDataInfo Value);
```

Description

Call GetDataInfo to get information about a data value associated with the current key. ValueName is a string containing the name of the data value to query. Value is a TRegDataInfo (see page 3327) value in which information about the data value is returned. If the data value is of type rdString or rdExpandString, the size of the data value includes an extra byte for the terminating null character.

GetDataInfo returns true if data value information is returned. If GetDataInfo fails, it returns false, and the Value record is filled with zeros.

See Also

CurrentKey (see page 3296)

GetDataSize (see page 3263)

GetDataType (see page 3264)

GetKeyInfo (see page 3264)

59.1.3.4.10 TRegistry.GetDataSize Method

Returns the size, in bytes, of a specified data value associated with the current key.

Pascal

```
function GetDataSize(const ValueName: string): Integer;
```

C++

```
__fastcall int GetDataSize(const UnicodeString ValueName);
```

Description

Call GetDataSize to determine the size, in bytes, of a data value associated with the current key. ValueName is a string containing the name of the data value to query.

On success, GetDataSize returns the size of the data value. On failure, GetDataSize returns -1.

Note: If the data value is a string, GetDataSize returns the size of the data value and one extra byte for the terminating null character.

See Also

CurrentKey (see page 3296)

GetDataInfo (see page 3263)

GetDataType (see page 3264)

GetKeyInfo (see page 3264)

59.1.3.4.11 TRegistry.GetDataType Method

Returns the enumerated data type of a specified data value associated with the current key.

Pascal

```
function GetDataType(const ValueName: string): TRegDataType;
```

C++

```
__fastcall TRegDataType GetDataType(const UnicodeString ValueName);
```

Description

Call GetDataType to determine the data type of a data value associated with the current key. ValueName is a string containing the name of the data value to query.

GetDataType returns a TRegDataType (see page 3327) value that indicates the type of value associated with the current key.

See Also

CurrentKey (see page 3296)

GetDataInfo (see page 3263)

GetDataSize (see page 3263)

GetKeyInfo (see page 3264)

59.1.3.4.12 TRegistry.GetKey Method

Pascal

```
function GetKey(const Key: string): HKEY;
```

C++

```
__fastcall HKEY GetKey(const UnicodeString Key);
```

Description

This is GetKey, a member of class TRegistry.

59.1.3.4.13 TRegistry.GetKeyInfo Method

Returns information about the current key.

Pascal

```
function GetKeyInfo(var Value: TRegKeyInfo): Boolean;
```

C++

```
__fastcall Boolean GetKeyInfo(TRegKeyInfo Value);
```

Description

Call GetKeyInfo to return all information about the current key to an application. Value is a variable declared in an application to store the key information returned by this function.

On success GetKeyInfo returns true. On failure it returns false, and the Value record is set to zeros.

See Also

CurrentKey (see page 3296)

GetKeyNames (see page 3265)

GetValueNames (see page 3265)

HasSubKeys (see page 3266)

59.1.3.4.14 TRegistry.GetKeyNames Method

Returns a string list containing the names of all subkeys belonging to the current key.

Pascal

```
procedure GetKeyNames(Strings: TStrings);
```

C++

```
__fastcall GetKeyNames(TStrings Strings);
```

Description

Call GetKeyNames to determine the names of all subkeys belonging to the current key. Determining the names of subkeys is useful in an application that iterates through a set of keys.

Strings is a variable of type TStrings into which to return the list of subkey names.

See Also

CurrentKey (see page 3296)

GetKeyInfo (see page 3264)

GetValueNames (see page 3265)

59.1.3.4.15 TRegistry.GetValueNames Method

Returns a string list containing the names of all data values associated with the current key.

Pascal

```
procedure GetValueNames(Strings: TStrings);
```

C++

```
__fastcall GetValueNames(TStrings Strings);
```

Description

Call GetValueNames to determine the names of all data values associated with the current key. Determining the names of data values is useful in an application that iterates through a set of data values for a key.

Strings is a variable of type TStrings into which to return the list of data value names.

See Also

CurrentKey (see page 3296)

GetDataInfo (see page 3263)

GetKeyNames (see page 3265)

ValueExists (see page 3281)

59.1.3.4.16 TRegistry.HasSubKeys Method

Determines if the current key has subkeys.

Pascal

```
function HasSubKeys: Boolean;
```

C++

```
__fastcall Boolean HasSubKeys();
```

Description

Call HasSubKeys to determine if the current key has any subkeys. HasSubKeys returns true if the current key has at least one subkey. Otherwise it returns false.

See Also

CurrentKey (see page 3296)

KeyExists (see page 3266)

59.1.3.4.17 TRegistry.KeyExists Method

Determines if a specified key exists.

Pascal

```
function KeyExists(const Key: string): Boolean;
```

C++

```
__fastcall Boolean KeyExists(const UnicodeString Key);
```

Description

Call KeyExists to determine if a key of a specified name exists. Key is the name of the key for which to search.

C++ Examples:

```
/*
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
for inserting and deleting the values.
*/
void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
{
    TRegistry* reg = new TRegistry(KEY_READ);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
    {
        MessageDlg("Key not found! Created now.",
                   mtInformation, TMsgDlgButtons() << mbOK, 0);
    }
    reg->Access = KEY_WRITE;
    bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);

    if(!openResult)
    {
        MessageDlg("Unable to create key! Exiting.",
                   mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }
```

```
    //checking if the values exist and inserting when neccesary

    if(!reg->KeyExists("Creation\ Date"))
    {
        TDateTime today = TDateTime::CurrentDateTime();
        reg->WriteDateTime("Creation\ Date", today);
    }

    if(!reg->KeyExists("Licenced\ To"))
    {
        reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
    }

    if(!reg->KeyExists("App\ Location"))
    {
        reg->WriteExpandString("App\ Location",
                               "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
    }

    if(!reg->KeyExists("Projects\ Location"))
    {
        reg->WriteExpandString("Projects\ Location",
                               "%USERPROFILE%\\MyApplication\\Projects\\");
    }

    reg->CloseKey();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
{
    //deleting the example registry entries
    TRegistry* reg = new TRegistry(KEY_WRITE);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
    reg->DeleteKey("Software\\MyCompanyName");

    reg->CloseKey();
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
on a form, for inserting and deleting the values.
}
procedure TForm3.InsertToRegBtnClick(Sender: TObject);
var
  reg        : TRegistry;
  openResult : Boolean;
  today      : TDateTime;
begin
  reg := TRegistry.Create(KEY_READ);
  reg.RootKey := HKEY_LOCAL_MACHINE;

  if (not reg.KeyExists('Software\\MyCompanyName\\MyApplication\\')) then
    begin
      MessageDlg('Key not found! Created now.',
                          mtInformation, mbOKCancel, 0);
    end;
  reg.Access := KEY_WRITE;
  openResult := reg.OpenKey('Software\\MyCompanyName\\MyApplication\\',True);

  if not openResult = True then
```

```
      begin
        MessageDlg('Unable to create key! Exiting.',
                   mtError, mbOKCancel, 0);
        Exit();
      end;

    {checking if the values exist and inserting when neccesary}

    if not reg.KeyExists('Creation\ Date') then
      begin
        today := Now;
          reg.WriteDateTime('Creation\ Date', today);
      end;

    if not reg.KeyExists('Licenced\ To') then
      begin
          reg.WriteString('Licenced\ To', 'MySurname\ MyFirstName');
      end;

    if not reg.KeyExists('App\ Location') then
      begin
          reg.WriteExpandString('App\ Location',
                                '%PROGRAMFILES%\\MyCompanyName\\MyApplication\\');
      end;

    if not reg.KeyExists('Projects\ Location') then
      begin
          reg.WriteExpandString('Projects\ Location',
                                '%USERPROFILE%\\MyApplication\\Projects\\');
      end;

    reg.CloseKey();

  end;

  procedure TForm3.DeleteFromRegBtnClick(Sender: TObject);
  var
    reg : TRegistry;
  begin
    reg := TRegistry.Create(KEY_WRITE);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    reg.DeleteKey('Software\\MyCompanyName\\MyApplication');
    reg.DeleteKey('Software\\MyCompanyName');

    reg.CloseKey();
  end;
```

See Also

LoadKey (see page 3268)

ValueExists (see page 3281)

59.1.3.4.18 TRegistry.LoadKey Method

Creates a subkey under the root key and loads registry information from a file into the newly created subkey.

Pascal

```
function LoadKey(const Key: string; const FileName: string): Boolean;
```

C++

```
__fastcall Boolean LoadKey(const UnicodeString Key, const UnicodeString FileName);
```

Description

Call LoadKey to:

1Create (see page 3251) a new subkey under the root key, and

2Load registry information from a file into the subkey. Registry (see page 3235) information can include data values, subkeys, and data values for those subkeys.

LoadKey is intended to simplify creation of a key, its values and subkeys, and the values for those subkeys in a single operation. A key, its subkeys, and all data values of the key and its subkeys is called a hive. Rather than creating each key and value separately, an application can read a hive from a file. This is especially useful for applications that users can reconfigure at run time.

Note: Before an application calls LoadKey, the RootKey (see page 3297) property must be set to HKEY_USERS, HKEY_LOCAL_MACHINE, or to a key returned by a previous call to RegistryConnect (see page 3277).

The Key parameter is the name of the subkey to create. The FileName parameter is the location of the file containing registry information to store in the subkey. The file specified by FileName must be one previously created using the SaveKey (see page 3279) function or the RegSaveKey Windows API function. On systems that use a file allocation table (FAT), FileName cannot include an extension.

See Also

RootKey (see page 3297)

CreateKey (see page 3258)

RegistryConnect (see page 3277)

SaveKey (see page 3279)

UnLoadKey (see page 3280)

59.1.3.4.19 TRegistry.MoveKey Method

Moves an existing key, its subkeys, and data values to a new location using a new key name.

Pascal

```
procedure MoveKey(const OldName: string; const NewName: string; Delete: Boolean);
```

C++

```
__fastcall MoveKey(const UnicodeString OldName, const UnicodeString NewName, Boolean Delete);
```

Description

Call MoveKey to copy or move an existing key, its subkeys, and data values to a different location. Under Windows 95, copying is recursive. If a key contains subkeys and data values, they too, are copied. When the operation is complete, the new key is closed. Under Windows NT, subkeys must be moved explicitly with separate calls to MoveKey.

OldName specifies the key to copy or move. NewName specifies a name for the duplicate key to create. If the key specified by NewName does not exist, MoveKey creates it. Delete specifies whether to delete the old key after the copy operation. If Delete is true, the key specified by OldName is deleted after it is copied. Otherwise the old key is closed.

See Also

CreateKey (see page 3258)

DeleteKey (see page 3259)

OpenKey (see page 3270)

59.1.3.4.20 TRegistry.OpenKey Method

Opens a specified key.

Pascal

```
function OpenKey(const Key: string; CanCreate: Boolean): Boolean;
```

C++

```
__fastcall Boolean OpenKey(const UnicodeString Key, Boolean CanCreate);
```

Description

Call OpenKey to make a specified key the current key. Key is the name of the key to open. If Key is nil (Delphi) or NULL (C++), the CurrentKey (see page 3296) property is set to the key specified by the RootKey (see page 3297) property.

CanCreate specifies whether to create the specified key if it does not exist. If CanCreate is true, the key is created if necessary.

Key is opened or created with the security access value specified by the Access (see page 3293) property. OpenKey only creates non-volatile keys, A non-volatile key is stored in the registry and is preserved when the system is restarted.

OpenKey returns true if the key is successfully opened or created

C++ Examples:

```
/*
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
for inserting and deleting the values.
*/
void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
{
    TRegistry* reg = new TRegistry(KEY_READ);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
    {
        MessageDlg("Key not found! Created now.",
                    mtInformation, TMsgDlgButtons() << mbOK, 0);
    }
    reg->Access = KEY_WRITE;
    bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);

    if(!openResult)
    {
        MessageDlg("Unable to create key! Exiting.",
                    mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }
    //checking if the values exist and inserting when neccesary

    if(!reg->KeyExists("Creation\ Date"))
    {
        TDateTime today = TDateTime::CurrentDateTime();
        reg->WriteDateTime("Creation\ Date", today);
    }

    if(!reg->KeyExists("Licenced\ To"))
    {
        reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
    }

    if(!reg->KeyExists("App\ Location"))
    {
```

```
        reg->WriteExpandString("App\ Location",
                               "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
    }

    if(!reg->KeyExists("Projects\ Location"))
    {
        reg->WriteExpandString("Projects\ Location",
                               "%USERPROFILE%\\MyApplication\\Projects\\");
    }

    reg->CloseKey();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
{
    //deleting the example registry entries
    TRegistry* reg = new TRegistry(KEY_WRITE);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
    reg->DeleteKey("Software\\MyCompanyName");

    reg->CloseKey();
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
on a form, for inserting and deleting the values.
}
procedure TForm3.InsertToRegBtnClick(Sender: TObject);
var
  reg        : TRegistry;
  openResult : Boolean;
  today      : TDateTime;
begin
  reg := TRegistry.Create(KEY_READ);
  reg.RootKey := HKEY_LOCAL_MACHINE;

  if (not reg.KeyExists('Software\\MyCompanyName\\MyApplication\\')) then
    begin
      MessageDlg('Key not found! Created now.',
                          mtInformation, mbOKCancel, 0);
    end;
  reg.Access := KEY_WRITE;
  openResult := reg.OpenKey('Software\\MyCompanyName\\MyApplication\\',True);

  if not openResult = True then
    begin
      MessageDlg('Unable to create key! Exiting.',
                 mtError, mbOKCancel, 0);
      Exit();
    end;

  {checking if the values exist and inserting when neccesary}

  if not reg.KeyExists('Creation\ Date') then
    begin
      today := Now;
        reg.WriteDateTime('Creation\ Date', today);
    end;

  if not reg.KeyExists('Licenced\ To') then
```

```
      begin
          reg.WriteString('Licenced\ To', 'MySurname\ MyFirstName');
      end;

    if not reg.KeyExists('App\ Location') then
      begin
          reg.WriteExpandString('App\ Location',
                                '%PROGRAMFILES%\\MyCompanyName\\MyApplication\\');
      end;

    if not reg.KeyExists('Projects\ Location') then
      begin
          reg.WriteExpandString('Projects\ Location',
                                '%USERPROFILE%\\MyApplication\\Projects\\');
      end;

    reg.CloseKey();

  end;

  procedure TForm3.DeleteFromRegBtnClick(Sender: TObject);
  var
    reg : TRegistry;
  begin
    reg := TRegistry.Create(KEY_WRITE);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    reg.DeleteKey('Software\\MyCompanyName\\MyApplication');
    reg.DeleteKey('Software\\MyCompanyName');

    reg.CloseKey();
  end;
```

See Also

CurrentKey (see page 3296)

RootKey (see page 3297)

CreateKey (see page 3258)

OpenKeyReadOnly (see page 3272)

Access (see page 3293)

59.1.3.4.21 TRegistry.OpenKeyReadOnly Method

Opens a specified key in a read-only manner.

Pascal

```
function OpenKeyReadOnly(const Key: String): Boolean;
```

C++

```
__fastcall Boolean OpenKeyReadOnly(const UnicodeString Key);
```

Description

Call OpenKeyReadOnly to make a specified key the current key. Key is the name of the key to open. If Key is nil (Delphi) or NULL (C++), the CurrentKey (see page 3296) property is set to the key specified by the RootKey (see page 3297) property.

Key is opened with the security access value KEY_READ.

OpenKey (see page 3270) returns true if the key is successfully opened and CurrentKey (see page 3296) is set to the key.

Note: After a successful call to OpenKeyReadOnly the Access (see page 3293) property of the registry component is

automatically changed to KEY_READ, regardless of what that property was set to prior to the method call.

See Also

CurrentKey (see page 3296)

RootKey (see page 3297)

CreateKey (see page 3258)

OpenKey (see page 3270)

Access (see page 3293)

59.1.3.4.22 TRegistry.PutData Method

Pascal

```
procedure PutData(const Name: string; Buffer: Pointer; BufSize: Integer; RegData:
TRegDataType);
```

C++

```
__fastcall PutData(const UnicodeString Name, void * Buffer, int BufSize, TRegDataType RegData);
```

Description

This is PutData, a member of class TRegistry.

59.1.3.4.23 TRegistry.ReadBinaryData Method

Retrieves a binary value from a specified data value associated with the current key.

Pascal

```
function ReadBinaryData(const Name: string; var Buffer; BufSize: Integer): Integer;
```

C++

```
__fastcall int ReadBinaryData(const UnicodeString Name,  Buffer, int BufSize);
```

Description

Call ReadBinaryData to read a binary value from a specified data value associated with the current key. Name is the name of the data value to read. Buffer is the application variable into which to read the registry data. Buffer must be large enough to hold all of the data returned. BufSize specifies the size of Buffer.

If successful, ReadBinaryData writes the requested data into Buffer and returns the number of bytes written. If the Registry (see page 3235) entry contains a known type (such as a string), ReadBinaryData raises an exception.

Note: Binary data is typically a complex data structure (a record in Delphi or struct in C++). It might also be an icon or a bitmap although Microsoft recommends against storing graphics objects in the registry for performance reasons.

See Also

CurrentKey (see page 3296)

ReadBool (see page 3274)

ReadCurrency (see page 3274)

ReadDate (see page 3274)

ReadDateTime (see page 3275)

ReadFloat (see page 3275)

ReadInteger (see page 3276)

ReadString (see page 3276)

ReadTime (see page 3277)

59.1.3.4.24 TRegistry.ReadBool Method

Retrieves a Boolean value from a specified data value associated with the current key.

Pascal

```
function ReadBool(const Name: string): Boolean;
```

C++

```
__fastcall Boolean ReadBool(const UnicodeString Name);
```

Description

Call ReadBool to read a Boolean value from a specified data value associated with the current key. Name is the name of the data value to read.

If successful, ReadBool returns the requested Boolean value. If the Registry (see page 3235) entry does not contain a boolean or integer value, an exception is raised.

See Also

CurrentKey (see page 3296)

WriteBool (see page 3282)

59.1.3.4.25 TRegistry.ReadCurrency Method

Retrieves a currency value from a specified data value associated with the current key.

Pascal

```
function ReadCurrency(const Name: string): Currency;
```

C++

```
__fastcall Currency ReadCurrency(const UnicodeString Name);
```

Description

Call ReadCurrency to read a Currency value from a specified data value associated with the current key. Name is the name of the data value to read.

If successful, ReadCurrency returns a Currency value. If the Registry (see page 3235) entry does not contain a Currency value, an exception is raised.

See Also

CurrentKey (see page 3296)

WriteCurrency (see page 3282)

59.1.3.4.26 TRegistry.ReadDate Method

Retrieves a date value from a specified data value associated with the current key.

Pascal

```
function ReadDate(const Name: string): TDateTime;
```

C++

```
__fastcall TDateTime ReadDate(const UnicodeString Name);
```

Description

Call ReadDate to read a date value from a specified data value associated with the current key. Name is the name of the data value to read.

If successful, ReadDate returns a TDateTime value. The integral part of a TDateTime value is the number of days that have passed since 12/30/1899. The fractional part of a TDateTime value is the time of day.

If the Registry (see page 3235) entry does not contain a date or time value, an exception is raised.

Note: Only use ReadDate to read data values previously stored with the WriteDate (see page 3282) procedure.

See Also

CurrentKey (see page 3296)

WriteDate (see page 3282)

ReadDateTime (see page 3275)

59.1.3.4.27 TRegistry.ReadDateTime Method

Retrieves a date and time value from a specified data value associated with the current key.

Pascal

```
function ReadDateTime(const Name: string): TDateTime;
```

C++

```
__fastcall TDateTime ReadDateTime(const UnicodeString Name);
```

Description

Call ReadDateTime to read a datetime value from a specified data value associated with the current key. Name is the name of the data value to read.

If successful, ReadDateTime returns a TDateTime value. The integral part of a TDateTime value is the number of days that have passed since 12/30/1899. The fractional part of a TDateTime value is the time of day.

If the Registry (see page 3235) entry does not contain a date or time value, an exception is raised.

See Also

CurrentKey (see page 3296)

WriteDateTime (see page 3283)

ReadDate (see page 3274)

ReadTime (see page 3277)

59.1.3.4.28 TRegistry.ReadFloat Method

Retrieves a float value from a specified data value associated with the current key.

Pascal

```
function ReadFloat(const Name: string): Double;
```

C++

```
__fastcall Double ReadFloat(const UnicodeString Name);
```

Description

Call ReadFloat to read a float value from a specified data value associated with the current key. Name is the name of the data value to read.

If the Registry (see page 3235) entry does not contain a floating-point value, an exception is raised.

See Also

CurrentKey (see page 3296)

WriteFloat (see page 3288)

59.1.3.4.29 TRegistry.ReadInteger Method

Retrieves an integer value from a specified data value associated with the current key.

Pascal

```
function ReadInteger(const Name: string): Integer;
```

C++

```
__fastcall int ReadInteger(const UnicodeString Name);
```

Description

Call ReadInteger to read an integer value from a specified data value associated with the current key. Name is the name of the data value to read.

If the Registry (see page 3235) entry does not contain an integer, an exception is raised.

See Also

CurrentKey (see page 3296)

WriteInteger (see page 3289)

59.1.3.4.30 TRegistry.ReadString Method

Retrieves a string value from a specified data value associated with the current key.

Pascal

```
function ReadString(const Name: string): string;
```

C++

```
__fastcall UnicodeString ReadString(const UnicodeString Name);
```

Description

Call ReadString to read a string value from a specified data value associated with the current key. Name is the name of the data value to read.

If there is no Registry (see page 3235) entry with the specified name, ReadString returns an empty string. If the Registry (see page 3235) entry contains something other than a string, an exception is raised.

Note: ReadString retrieves data stored by either the WriteString (see page 3289) or WriteExpandString (see page 3286) method.

See Also

CurrentKey (see page 3296)

WriteString (see page 3289)

WriteExpandString (see page 3286)

59.1.3.4.31 TRegistry.ReadTime Method

Retrieves a time value from a specified data value associated with the current key.

Pascal

```
function ReadTime(const Name: string): TDateTime;
```

C++

```
__fastcall TDateTime ReadTime(const UnicodeString Name);
```

Description

Call ReadTime to read a time value from a specified data value associated with the current key. Name is the name of the data value to read.

If successful, ReadTime returns a TDateTime value. The integral part of a TDateTime value is the number of days that have passed since 12/30/1899. The fractional part of a TDateTime value is the time of day.

If the specified Registry (see page 3235) entry does not contain a date or time value, an exception is raised.

Note: Only use ReadTime to read data values previously stored with the WriteTime (see page 3292) method.

See Also

CurrentKey (see page 3296)

WriteTime (see page 3292)

ReadDateTime (see page 3275)

59.1.3.4.32 TRegistry.RegistryConnect Method

Establishes a connection to a registry on another computer.

Pascal

```
function RegistryConnect(const UNCName: string): Boolean;
```

C++

```
__fastcall Boolean RegistryConnect(const UnicodeString UNCName);
```

Description

Call RegistryConnect to establish a connection to a registry on another computer running Windows 95 or Windows NT. UNCName is the name of the remote computer, and must take the following form:

```
\\computername
```

where computername is the name of the remote computer. If UNCName is nil (Delphi) or NULL (C++), the local machine name is used.

Note: Before calling RegistryConnect, an application must set the RootKey (see page 3297) property for its registry object to HKEY_USERS or HKEY_LOCAL_MACHINE.

If connection is successful, RegistryConnect sets the RootKey (see page 3297) property to the root key of the remote

computer and returns true. If unsuccessful, RegistryConnect returns false, and RootKey (see page 3297) remains unchanged.

See Also

RootKey (see page 3297)

LoadKey (see page 3268)

ReplaceKey (see page 3278)

RestoreKey (see page 3279)

SaveKey (see page 3279)

59.1.3.4.33 TRegistry.RenameValue Method

Changes the name of a data value associated with the current key.

Pascal

```
procedure RenameValue(const OldName: string; const NewName: string);
```

C++

```
__fastcall RenameValue(const UnicodeString OldName, const UnicodeString NewName);
```

Description

Call RenameValue to change the name of a data value associated with the current key. OldName is a string containing the current name of the data value. NewName is a string containing the replacement name for the data value.

If OldName is the name of an existing data value for the current key, and NewName is not the name of an existing data value for the current key, RenameValue changes the data value name as specified. Otherwise the current name remains unchanged.

See Also

CurrentKey (see page 3296)

GetValueNames (see page 3265)

ValueExists (see page 3281)

MoveKey (see page 3269)

59.1.3.4.34 TRegistry.ReplaceKey Method

Replaces the hive file backing a key and all its subkeys with a hive stored in a different file, so that when the system is next started, the key and subkeys will have the values stored in the new file.

Pascal

```
function ReplaceKey(const Key: string; const FileName: string; const BackUpFileName: string):
Boolean;
```

C++

```
__fastcall Boolean ReplaceKey(const UnicodeString Key, const UnicodeString FileName, const
UnicodeString BackUpFileName);
```

Description

Call ReplaceKey to replace the hive file backing a key and all its subkeys with a different hive file. A hive is a discrete collection of keys, subkeys, and values that is rooted at the top of the registry hierarchy.

FileName indicates the name of the file containing the new key and subkey information. BackUpFileName is the name of a backup file used to store the current key and subkey information before it is overwritten.

ReplaceKey returns true if it is successful.

Note: The file specified by FileName parameter must be created with the SaveKey (see page 3279) function, or by directly calling the Windows 95/NT RegSaveKey API function. Under the FAT file system FileName cannot have an extension.

See Also

LoadKey (see page 3268)

RestoreKey (see page 3279)

SaveKey (see page 3279)

59.1.3.4.35 TRegistry.RestoreKey Method

Reopens the specified key, and overwrites any information contained in this key and its subkeys keys with registry data stored in a hive file.

Pascal

```
function RestoreKey(const Key: string; const FileName: string): Boolean;
```

C++

```
__fastcall Boolean RestoreKey(const UnicodeString Key, const UnicodeString FileName);
```

Description

Call RestoreKey to open the specified key with a security access value of KEY_ALL_ACCESS and overwrite any information in the key and its subkeys with registry data stored in a hive file. A hive is a discrete collection of keys, subkeys, and values that is rooted at the top of the registry hierarchy.

Key is the name of the key to open. FileName is the name of the file containing the hive information.

If RestoreKey successfully reads and copies the registry information in FileName to the specified key, the return value of RestoreKey is true and the key is closed.

Note: The file specified by FileName parameter must be created with the SaveKey (see page 3279) function, or by directly calling the Windows 95/NT RegSaveKey API function. Under the FAT file system FileName cannot have an extension.

See Also

LoadKey (see page 3268)

ReplaceKey (see page 3278)

RegistryConnect (see page 3277)

SaveKey (see page 3279)

59.1.3.4.36 TRegistry.SaveKey Method

Opens the specified key with the security access value KEY_ALL_ACCESS and saves the specified key and all of its subkeys and values to a hive file.

Pascal

```
function SaveKey(const Key: string; const FileName: string): Boolean;
```

C++

```
__fastcall Boolean SaveKey(const UnicodeString Key, const UnicodeString FileName);
```

Description

Call SaveKey to open a key with a security access value of KEY_ALL_ACCESS, and save the key and its subkeys and data values to a hive file. A hive is a discrete collection of keys, subkeys, and values that is rooted at the top of the registry hierarchy.

Key is the key to save. FileName is the file into which to save the key information. It must be the name of a new file that does not already exist. On FAT file systems FileName cannot include an extension.

Files created by SaveKey are passed as parameters to the RestoreKey (see page 3279), ReplaceKey (see page 3278) and LoadKey (see page 3268) functions.

If SaveKey is successful it returns true and closes the key.

See Also

LoadKey (see page 3268)

ReplaceKey (see page 3278)

RestoreKey (see page 3279)

59.1.3.4.37 TRegistry.SetCurrentKey Method

Pascal

```
procedure SetCurrentKey(Value: HKEY);
```

C++

```
__fastcall SetCurrentKey(HKEY Value);
```

Description

This is SetCurrentKey, a member of class TRegistry.

59.1.3.4.38 TRegistry.UnLoadKey Method

Removes a hive from the registry.

Pascal

```
function UnLoadKey(const Key: string): Boolean;
```

C++

```
__fastcall Boolean UnLoadKey(const UnicodeString Key);
```

Description

Call UnLoadKey to remove a hive from the registry without modifying its associated hive file, if any. A hive is a discrete collection of keys, subkeys, and values that is rooted at the top of the registry hierarchy.

Key is the name of the key at the root of the hive to remove.

If successful, UnLoadKey returns true.

Note: Before calling this function, set RootKey (see page 3297) to HKEY_USERS, HKEY_LOCAL_MACHINE, or call RegistryConnect (see page 3277).

See Also

RootKey (see page 3297)

LoadKey (see page 3268)

ReplaceKey (see page 3278)

RestoreKey (see page 3279)

SaveKey (see page 3279)

59.1.3.4.39 TRegistry.ValueExists Method

Determines if a specified data value exists for the current key.

Pascal

```
function ValueExists(const Name: string): Boolean;
```

C++

```
__fastcall Boolean ValueExists(const UnicodeString Name);
```

Description

Call ValueExists to determine if a particular data item exists in the registry. Calling ValueExists is especially useful before calling other TRegistry (see page 3246) methods that operate only on existing data items.

Name is the name of the data value for which to check.

ValueExists returns true if a match if found, false otherwise.

See Also

CurrentKey (see page 3296)

KeyExists (see page 3266)

59.1.3.4.40 TRegistry.WriteBinaryData Method

Stores a specifically named data value associated with the current key.

Pascal

```
procedure WriteBinaryData(const Name: string; var Buffer; BufSize: Integer);
```

C++

```
__fastcall WriteBinaryData(const UnicodeString Name,  Buffer, int BufSize);
```

Description

Call WriteBinaryData to store a data value associated with the current key.

Name is a string containing the name of the data value in which to store data. If Name already exists, its current value is overwritten by WriteBinary. If Name does not exist, it is created.

Buffer is a data buffer containing the data to store in the registry. BufSize indicates the size of Buffer.

Note: Data-value lengths are limited by available memory. Long values (more than 2048 bytes) should be stored as separate files, and those file names should be stored in the registry. Application elements such as icons, bitmaps, and executable files should be stored as separate files rather than as values in the registry.

If WriteBinaryData fails, an exception is raised, and the value is not written to the registry.

See Also

CurrentKey (see page 3296)

ReadBinaryData (see page 3273)

59.1.3.4.41 TRegistry.WriteBool Method

Stores a Boolean value in a specifically named data value associated with the current key.

Pascal

```
procedure WriteBool(const Name: string; Value: Boolean);
```

C++

```
__fastcall WriteBool(const UnicodeString Name, Boolean Value);
```

Description

Call WriteBool to store a boolean value in a data value associated with the current key.

Name is a string containing the name of the data value in which to store data. If Name already exists, its current value is overwritten by WriteBool. If Name does not exist, it is created.

If WriteBool fails, an exception is raised, and the value is not written to the registry.

See Also

CurrentKey (see page 3296)

ReadBool (see page 3274)

59.1.3.4.42 TRegistry.WriteCurrency Method

Stores a currency value in a specifically named data value associated with the current key.

Pascal

```
procedure WriteCurrency(const Name: string; Value: Currency);
```

C++

```
__fastcall WriteCurrency(const UnicodeString Name, Currency Value);
```

Description

Call WriteCurrency to store a Currency value in a data value associated with the current key.

Name is a string containing the name of the data value in which to store data. If Name already exists, its current value is overwritten by WriteCurrency. If Name does not exist, it is created.

Value is a Currency value to store in the registry.

If WriteCurrency fails, an exception is raised, and the value is not written to the registry.

See Also

CurrentKey (see page 3296)

ReadCurrency (see page 3274)

59.1.3.4.43 TRegistry.WriteDate Method

Stores a date value in a specifically named data value associated with the current key.

Pascal

```
procedure WriteDate(const Name: string; Value: TDateTime);
```

C++

```
__fastcall WriteDate(const UnicodeString Name, TDateTime Value);
```

Description

Call WriteDate to store a datetime value in a data value associated with the current key.

Name is a string containing the name of the data value in which to store data. If Name already exists, its current value is overwritten by WriteDate. If Name does not exist, it is created.

Value is a datetime value to store in the registry.

If WriteDate fails, an exception is raised, and the value is not written to the registry.

If successful, WriteDate stores a TDateTime value. The integral part of a TDateTime value is the number of days that have passed since 12/30/1899. The fractional part of a TDateTime value is the time of day.

See Also

CurrentKey (see page 3296)

ReadDate (see page 3274)

WriteDateTime (see page 3283)

59.1.3.4.44 TRegistry.WriteDateTime Method

Stores a datetime value in a specifically named data value associated with the current key.

Pascal

```
procedure WriteDateTime(const Name: string; Value: TDateTime);
```

C++

```
__fastcall WriteDateTime(const UnicodeString Name, TDateTime Value);
```

Description

Call WriteDateTime to store a datetime value in a data value associated with the current key.

Name is a string containing the name of the data value in which to store data. If Name already exists, its current value is overwritten by WriteDateTime. If Name does not exist, it is created.

Value is a datetime value to store in the registry.

If WriteDateTime fails, an exception is raised, and the value is not written to the registry.

If successful, WriteDateTime stores a TDateTime value. The integral part of a TDateTime value is the number of days that have passed since 12/30/1899. The fractional part of a TDateTime value is the time of day.

C++ Examples:

```
/*
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
for inserting and deleting the values.
*/
void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
{
    TRegistry* reg = new TRegistry(KEY_READ);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
    {
```

```
        MessageDlg("Key not found! Created now.",
                   mtInformation, TMsgDlgButtons() << mbOK, 0);
    }
    reg->Access = KEY_WRITE;
    bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);

    if(!openResult)
    {
        MessageDlg("Unable to create key! Exiting.",
                   mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }
    //checking if the values exist and inserting when neccesary

    if(!reg->KeyExists("Creation\ Date"))
    {
        TDateTime today = TDateTime::CurrentDateTime();
        reg->WriteDateTime("Creation\ Date", today);
    }

    if(!reg->KeyExists("Licenced\ To"))
    {
        reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
    }

    if(!reg->KeyExists("App\ Location"))
    {
        reg->WriteExpandString("App\ Location",
                               "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
    }

    if(!reg->KeyExists("Projects\ Location"))
    {
        reg->WriteExpandString("Projects\ Location",
                               "%USERPROFILE%\\MyApplication\\Projects\\");
    }

    reg->CloseKey();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
{
    //deleting the example registry entries
    TRegistry* reg = new TRegistry(KEY_WRITE);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
    reg->DeleteKey("Software\\MyCompanyName");

    reg->CloseKey();
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
on a form, for inserting and deleting the values.
}
procedure TForm3.InsertToRegBtnClick(Sender: TObject);
var
  reg        : TRegistry;
  openResult : Boolean;
  today      : TDateTime;
begin
```

```
    reg := TRegistry.Create(KEY_READ);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    if (not reg.KeyExists('Software\\MyCompanyName\\MyApplication\\')) then
      begin
        MessageDlg('Key not found! Created now.',
                               mtInformation, mbOKCancel, 0);
      end;
    reg.Access := KEY_WRITE;
    openResult := reg.OpenKey('Software\\MyCompanyName\\MyApplication\\',True);

    if not openResult = True then
      begin
        MessageDlg('Unable to create key! Exiting.',
                    mtError, mbOKCancel, 0);
        Exit();
      end;

    {checking if the values exist and inserting when neccesary}

    if not reg.KeyExists('Creation\ Date') then
      begin
        today := Now;
          reg.WriteDateTime('Creation\ Date', today);
      end;

    if not reg.KeyExists('Licenced\ To') then
      begin
          reg.WriteString('Licenced\ To', 'MySurname\ MyFirstName');
      end;

    if not reg.KeyExists('App\ Location') then
      begin
          reg.WriteExpandString('App\ Location',
                                '%PROGRAMFILES%\\MyCompanyName\\MyApplication\\');
      end;

    if not reg.KeyExists('Projects\ Location') then
      begin
          reg.WriteExpandString('Projects\ Location',
                                '%USERPROFILE%\\MyApplication\\Projects\\');
      end;

    reg.CloseKey();

  end;

  procedure TForm3.DeleteFromRegBtnClick(Sender: TObject);
  var
    reg : TRegistry;
  begin
    reg := TRegistry.Create(KEY_WRITE);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    reg.DeleteKey('Software\\MyCompanyName\\MyApplication');
    reg.DeleteKey('Software\\MyCompanyName');

    reg.CloseKey();
  end;
```

See Also

CurrentKey (see page 3296)

ReadDateTime (see page 3275)

WriteDate (see page 3282)

WriteTime (see page 3292)

59.1.3.4.45 TRegistry.WriteExpandString Method

Stores a string containing unexpanded references to environment variables in a specifically named data value associated with the current key.

Pascal

```
procedure WriteExpandString(const Name: string; const Value: string);
```

C++

```
__fastcall WriteExpandString(const UnicodeString Name, const UnicodeString Value);
```

Description

Call WriteExpandString to store a string that contains unexpanded references to environment variables such as "%PATH%". The string is stored in a data value associated with the current key.

Name is a string containing the name of the data value in which to store data. If Name already exists, its current value is overwritten by WriteExpandString. If Name does not exist, it is created.

Value is a string value to store in the registry.

If WriteExpandString fails, an exception is raised.

Note: To write strings that do not contain unexpanded environment variables, use WriteString (see page 3289) instead.

C++ Examples:

```
/*
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
for inserting and deleting the values.
*/
void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
{
    TRegistry* reg = new TRegistry(KEY_READ);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
    {
        MessageDlg("Key not found! Created now.",
                   mtInformation, TMsgDlgButtons() << mbOK, 0);
    }
    reg->Access = KEY_WRITE;
    bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);

    if(!openResult)
    {
        MessageDlg("Unable to create key! Exiting.",
                   mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }
    //checking if the values exist and inserting when neccesary

    if(!reg->KeyExists("Creation\ Date"))
    {
        TDateTime today = TDateTime::CurrentDateTime();
        reg->WriteDateTime("Creation\ Date", today);
    }
```

```
    if(!reg->KeyExists("Licenced\ To"))
    {
        reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
    }

    if(!reg->KeyExists("App\ Location"))
    {
        reg->WriteExpandString("App\ Location",
                               "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
    }

    if(!reg->KeyExists("Projects\ Location"))
    {
        reg->WriteExpandString("Projects\ Location",
                               "%USERPROFILE%\\MyApplication\\Projects\\");
    }

    reg->CloseKey();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
{
    //deleting the example registry entries
    TRegistry* reg = new TRegistry(KEY_WRITE);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
    reg->DeleteKey("Software\\MyCompanyName");

    reg->CloseKey();
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
on a form, for inserting and deleting the values.
}
procedure TForm3.InsertToRegBtnClick(Sender: TObject);
var
  reg        : TRegistry;
  openResult : Boolean;
  today      : TDateTime;
begin
  reg := TRegistry.Create(KEY_READ);
  reg.RootKey := HKEY_LOCAL_MACHINE;

  if (not reg.KeyExists('Software\\MyCompanyName\\MyApplication\\')) then
    begin
      MessageDlg('Key not found! Created now.',
                          mtInformation, mbOKCancel, 0);
    end;
  reg.Access := KEY_WRITE;
  openResult := reg.OpenKey('Software\\MyCompanyName\\MyApplication\\',True);

  if not openResult = True then
    begin
      MessageDlg('Unable to create key! Exiting.',
                  mtError, mbOKCancel, 0);
      Exit();
    end;

  {checking if the values exist and inserting when neccesary}
```

```
    if not reg.KeyExists('Creation\ Date') then
      begin
        today := Now;
          reg.WriteDateTime('Creation\ Date', today);
      end;

    if not reg.KeyExists('Licenced\ To') then
      begin
          reg.WriteString('Licenced\ To', 'MySurname\ MyFirstName');
      end;

    if not reg.KeyExists('App\ Location') then
      begin
          reg.WriteExpandString('App\ Location',
                                '%PROGRAMFILES%\\MyCompanyName\\MyApplication\\');
      end;

    if not reg.KeyExists('Projects\ Location') then
      begin
          reg.WriteExpandString('Projects\ Location',
                                '%USERPROFILE%\\MyApplication\\Projects\\');
      end;

    reg.CloseKey();

  end;

  procedure TForm3.DeleteFromRegBtnClick(Sender: TObject);
  var
    reg : TRegistry;
  begin
    reg := TRegistry.Create(KEY_WRITE);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    reg.DeleteKey('Software\\MyCompanyName\\MyApplication');
    reg.DeleteKey('Software\\MyCompanyName');

    reg.CloseKey();
  end;
```

See Also

CurrentKey (see page 3296)

ReadString (see page 3276)

WriteString (see page 3289)

59.1.3.4.46 TRegistry.WriteFloat Method

Stores a Double value in a specifically named data value associated with the current key.

Pascal

```
procedure WriteFloat(const Name: string; Value: Double);
```

C++

```
__fastcall WriteFloat(const UnicodeString Name, Double Value);
```

Description

Call WriteFloat to store a float value in a data value associated with the current key.

Name is a string containing the name of the data value in which to store data. If Name already exists, its current value is overwritten by WriteFloat. If Name does not exist, it is created.

Value is a float value to store in the registry.

If WriteFloat fails, an exception is raised, and the value is not written to the registry.

See Also

CurrentKey (see page 3296)

ReadFloat (see page 3275)

59.1.3.4.47 TRegistry.WriteInteger Method

Stores an integer value in a specifically named data value associated with the current key.

Pascal

```
procedure WriteInteger(const Name: string; Value: Integer);
```

C++

```
__fastcall WriteInteger(const UnicodeString Name, int Value);
```

Description

Call WriteInteger to store an integer value in a data value associated with the current key.

Name is a string containing the name of the data value in which to store data. If Name already exists, its current value is overwritten by WriteInteger. If Name does not exist, it is created.

Value is a integer value to store in the registry.

If WriteInteger fails, an exception is raised, and the value is not written to the registry.

See Also

CurrentKey (see page 3296)

ReadInteger (see page 3276)

59.1.3.4.48 TRegistry.WriteString Method

Stores a string value in a specifically named data value associated with the current key.

Pascal

```
procedure WriteString(const Name: string; const Value: string);
```

C++

```
__fastcall WriteString(const UnicodeString Name, const UnicodeString Value);
```

Description

Call WriteString to store a string value in a data value associated with the current key.

Name is a string containing the name of the data value in which to store data. If Name already exists, its current value is overwritten by WriteString. If Name does not exist, it is created.

Value is a string value to store in the registry.

If WriteString fails, an exception is raised, and the value is not written to the registry.

C++ Examples:

```
/*
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
```

```
  This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
  for inserting and deleting the values.
  */
  void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
  {
      TRegistry* reg = new TRegistry(KEY_READ);
      reg->RootKey = HKEY_LOCAL_MACHINE;

      if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
      {
          MessageDlg("Key not found! Created now.",
                     mtInformation, TMsgDlgButtons() << mbOK, 0);
      }
      reg->Access = KEY_WRITE;
      bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);

      if(!openResult)
      {
          MessageDlg("Unable to create key! Exiting.",
                     mtError, TMsgDlgButtons() << mbOK, 0);
          return;
      }
      //checking if the values exist and inserting when neccesary

      if(!reg->KeyExists("Creation\ Date"))
      {
          TDateTime today = TDateTime::CurrentDateTime();
          reg->WriteDateTime("Creation\ Date", today);
      }

      if(!reg->KeyExists("Licenced\ To"))
      {
          reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
      }

      if(!reg->KeyExists("App\ Location"))
      {
          reg->WriteExpandString("App\ Location",
                                 "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
      }

      if(!reg->KeyExists("Projects\ Location"))
      {
          reg->WriteExpandString("Projects\ Location",
                                 "%USERPROFILE%\\MyApplication\\Projects\\");
      }

      reg->CloseKey();
  }
  //---------------------------------------------------------------------------
  void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
  {
      //deleting the example registry entries
      TRegistry* reg = new TRegistry(KEY_WRITE);
      reg->RootKey = HKEY_LOCAL_MACHINE;

      reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
      reg->DeleteKey("Software\\MyCompanyName");

      reg->CloseKey();
  }
  //---------------------------------------------------------------------------
```

Delphi Examples:

```
  {
  This example shows how to use the TRegistry class in order to
```

```
  find, insert and delete Keys and Items into the Windows Registry.
  This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
  on a form, for inserting and deleting the values.
  }
  procedure TForm3.InsertToRegBtnClick(Sender: TObject);
  var
    reg        : TRegistry;
    openResult : Boolean;
    today      : TDateTime;
  begin
    reg := TRegistry.Create(KEY_READ);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    if (not reg.KeyExists('Software\\MyCompanyName\\MyApplication\\')) then
      begin
        MessageDlg('Key not found! Created now.',
                              mtInformation, mbOKCancel, 0);
      end;
    reg.Access := KEY_WRITE;
    openResult := reg.OpenKey('Software\\MyCompanyName\\MyApplication\\',True);

    if not openResult = True then
      begin
        MessageDlg('Unable to create key! Exiting.',
                    mtError, mbOKCancel, 0);
        Exit();
      end;

    {checking if the values exist and inserting when neccesary}

    if not reg.KeyExists('Creation\ Date') then
      begin
        today := Now;
          reg.WriteDateTime('Creation\ Date', today);
      end;

    if not reg.KeyExists('Licenced\ To') then
      begin
          reg.WriteString('Licenced\ To', 'MySurname\ MyFirstName');
      end;

    if not reg.KeyExists('App\ Location') then
      begin
          reg.WriteExpandString('App\ Location',
                                '%PROGRAMFILES%\\MyCompanyName\\MyApplication\\');
      end;

    if not reg.KeyExists('Projects\ Location') then
      begin
          reg.WriteExpandString('Projects\ Location',
                                '%USERPROFILE%\\MyApplication\\Projects\\');
      end;

    reg.CloseKey();

  end;

  procedure TForm3.DeleteFromRegBtnClick(Sender: TObject);
  var
    reg : TRegistry;
  begin
    reg := TRegistry.Create(KEY_WRITE);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    reg.DeleteKey('Software\\MyCompanyName\\MyApplication');
    reg.DeleteKey('Software\\MyCompanyName');
```

```
  reg.CloseKey();
end;
```

See Also

CurrentKey (see page 3296)

ReadString (see page 3276)

WriteExpandString (see page 3286)

59.1.3.4.49 TRegistry.WriteTime Method

Stores a time value in a specifically named data value associated with the current key.

Pascal

```
procedure WriteTime(const Name: string; Value: TDateTime);
```

C++

```
__fastcall WriteTime(const UnicodeString Name, TDateTime Value);
```

Description

Call WriteTime to store a datetime value in a data value associated with the current key.

Name is a string containing the name of the data value in which to store data. If Name already exists, its current value is overwritten by WriteTime. If Name does not exist, it is created.

Value is a datetime value to store in the registry.

If WriteTime fails, an exception is raised, and the value is not written to the registry.

If successful, WriteTime stores a TDateTime value. The integral part of a TDateTime value is the number of days that have passed since 12/30/1899. The fractional part of a TDateTime value is the time of day.

See Also

CurrentKey (see page 3296)

ReadTime (see page 3277)

WriteDateTime (see page 3283)

59.1.3.5 TRegistry Properties

The properties of the TRegistry class are listed here.

Public Properties

	Name	Description
	Access (see page 3293)	Specifies the level of access to use when opening keys.
R	CurrentKey (see page 3296)	Specifies the registry key that is currently open.
R	CurrentPath (see page 3296)	Indicates the registry path associated with the current key.
	LazyWrite (see page 3296)	Specifies how keys are written to the registry when calling the CloseKey (see page 3255) procedure.
	RootKey (see page 3297)	Specifies the root key for the TRegistry (see page 3246) object in an application.

59.1.3.5.1 TRegistry.Access Property

Specifies the level of access to use when opening keys.

Pascal

```
property Access: LongWord;
```

C++

```
__property LongWord Access;
```

Description

Use Access to specify the level of security access to use when opening keys. The OpenKey (see page 3270) method uses the value of Access when opening a registry key. Access is initialized by the TRegistry (see page 3246) constructor, but can be altered before calling OpenKey (see page 3270).

The value of Access is a set of flags or'd together, where each flag indicates a type of permission. Choose a value that grants sufficient permission to allow the desired actions while not exceeding the access level of the system running the application. The following table lists the flags that can be included:

Flag	Meaning
KEY_ALL_ACCESS	combination of KEY_READ, KEY_WRITE, and KEY_CREATE_LINK.
KEY_READ	combination of KEY_QUERY_VALUE, KEY_ENUMERATE_SUB_KEYS, and KEY_NOTIFY.
KEY_WRITE	combination of KEY_SET_VALUE and KEY_CREATE_SUB_KEY.
KEY_QUERY_VALUE	grants permission to query subkey data.
KEY_ENUMERATE_SUB_KEYS	grants permission to enumerate subkeys.
KEY_NOTIFY	grants ability to receive change notifications.
KEY_SET_VALUE	grants permission to set subkey data.
KEY_CREATE_SUB_KEY	grants permission to create subkeys.
KEY_CREATE_LINK	grants permission to create symbolic links.
KEY_EXECUTE	restricts permission to read-only access.

Note: Access is ignored by the CreateKey (see page 3258) method, which always uses KEY_ALL_ACCESS.

C++ Examples:

```
/*
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
for inserting and deleting the values.
*/
void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
{
    TRegistry* reg = new TRegistry(KEY_READ);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
    {
        MessageDlg("Key not found! Created now.",
                   mtInformation, TMsgDlgButtons() << mbOK, 0);
    }
```

```
    reg->Access = KEY_WRITE;
    bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);

    if(!openResult)
    {
        MessageDlg("Unable to create key! Exiting.",
                   mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }
    //checking if the values exist and inserting when neccesary

    if(!reg->KeyExists("Creation\ Date"))
    {
        TDateTime today = TDateTime::CurrentDateTime();
        reg->WriteDateTime("Creation\ Date", today);
    }

    if(!reg->KeyExists("Licenced\ To"))
    {
        reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
    }

    if(!reg->KeyExists("App\ Location"))
    {
        reg->WriteExpandString("App\ Location",
                               "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
    }

    if(!reg->KeyExists("Projects\ Location"))
    {
        reg->WriteExpandString("Projects\ Location",
                               "%USERPROFILE%\\MyApplication\\Projects\\");
    }

    reg->CloseKey();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
{
    //deleting the example registry entries
    TRegistry* reg = new TRegistry(KEY_WRITE);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
    reg->DeleteKey("Software\\MyCompanyName");

    reg->CloseKey();
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
on a form, for inserting and deleting the values.
}
procedure TForm3.InsertToRegBtnClick(Sender: TObject);
var
  reg        : TRegistry;
  openResult : Boolean;
  today      : TDateTime;
begin
  reg := TRegistry.Create(KEY_READ);
  reg.RootKey := HKEY_LOCAL_MACHINE;
```

```
    if (not reg.KeyExists('Software\\MyCompanyName\\MyApplication\\')) then
      begin
        MessageDlg('Key not found! Created now.',
                               mtInformation, mbOKCancel, 0);
      end;
    reg.Access := KEY_WRITE;
    openResult := reg.OpenKey('Software\\MyCompanyName\\MyApplication\\',True);

    if not openResult = True then
      begin
        MessageDlg('Unable to create key! Exiting.',
                     mtError, mbOKCancel, 0);
        Exit();
      end;

    {checking if the values exist and inserting when neccesary}

    if not reg.KeyExists('Creation\ Date') then
      begin
        today := Now;
          reg.WriteDateTime('Creation\ Date', today);
      end;

    if not reg.KeyExists('Licenced\ To') then
      begin
          reg.WriteString('Licenced\ To', 'MySurname\ MyFirstName');
      end;

    if not reg.KeyExists('App\ Location') then
      begin
          reg.WriteExpandString('App\ Location',
                               '%PROGRAMFILES%\\MyCompanyName\\MyApplication\\');
      end;

    if not reg.KeyExists('Projects\ Location') then
      begin
          reg.WriteExpandString('Projects\ Location',
                               '%USERPROFILE%\\MyApplication\\Projects\\');
      end;

    reg.CloseKey();

  end;

  procedure TForm3.DeleteFromRegBtnClick(Sender: TObject);
  var
    reg : TRegistry;
  begin
    reg := TRegistry.Create(KEY_WRITE);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    reg.DeleteKey('Software\\MyCompanyName\\MyApplication');
    reg.DeleteKey('Software\\MyCompanyName');

    reg.CloseKey();
  end;
```

See Also

OpenKey (see page 3270)

Create (see page 3251)

OpenKeyReadOnly (see page 3272)

59.1.3.5.2 TRegistry.CurrentKey Property

Specifies the registry key that is currently open.

Pascal

```
property CurrentKey: HKEY;
```

C++

```
__property HKEY CurrentKey;
```

Description

Use CurrentKey to determine the registry key that is currently open. All operations performed by TRegistry (see page 3246) affect only the current key. To access another key, call the OpenKey (see page 3270) or OpenKeyReadOnly (see page 3272) method.

See Also

CurrentPath (see page 3296)

RootKey (see page 3297)

OpenKey (see page 3270)

OpenKeyReadOnly (see page 3272)

59.1.3.5.3 TRegistry.CurrentPath Property

Indicates the registry path associated with the current key.

Pascal

```
property CurrentPath: string;
```

C++

```
__property UnicodeString CurrentPath;
```

Description

Use CurrentPath to determine the registry path of the current key. The system registry is hierarchical, and CurrentPath returns a string that spells out the hierarchy of keys from the root key to the current key, inclusive.

CurrentPath is automatically reset each time a different key becomes the current key.

See Also

CurrentKey (see page 3296)

RootKey (see page 3297)

59.1.3.5.4 TRegistry.LazyWrite Property

Specifies how keys are written to the registry when calling the CloseKey (see page 3255) procedure.

Pascal

```
property LazyWrite: Boolean;
```

C++

```
__property Boolean LazyWrite;
```

Description

Use LazyWrite to specify how keys are written to the registry. LazyWrite is initialized to true when a registry object is first created. When LazyWrite is true, keys are written to the registry when they are closed, but the CloseKey (see page 3255) procedure may return before the write operation takes place.

When LazyWrite is false, keys are written to the registry before CloseKey (see page 3255) returns. Set LazyWrite to false only when absolutely necessary. Setting LazyWrite to false guarantees that the changes an application makes to keys are written to the registry before further application processing takes place, but it uses more system resources when keys are closed. If LazyWrite is false, and an application is displaying or changing many registry entries, there will be a degradation in application performance.

See Also

CloseKey (see page 3255)

59.1.3.5.5 TRegistry.RootKey Property

Specifies the root key for the TRegistry (see page 3246) object in an application.

Pascal

```
property RootKey: HKEY;
```

C++

```
__property HKEY RootKey;
```

Description

Use RootKey to determine the hierarchy of subkeys an application can access, or to specify the root key for the application.

By default, RootKey is set to HKEY_CURRENT_USER when the registry object is created. To change the root key, specify a valid integer value for the RootKey property.

Note: For some operations, such as LoadKey (see page 3268) or RegistryConnect (see page 3277), an application may need to change the default value of RootKey.

C++ Examples:

```
/*
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
for inserting and deleting the values.
*/
void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
{
    TRegistry* reg = new TRegistry(KEY_READ);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
    {
        MessageDlg("Key not found! Created now.",
                    mtInformation, TMsgDlgButtons() << mbOK, 0);
    }
    reg->Access = KEY_WRITE;
    bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);

    if(!openResult)
    {
        MessageDlg("Unable to create key! Exiting.",
```

```
                    mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }
    //checking if the values exist and inserting when neccesary

    if(!reg->KeyExists("Creation\ Date"))
    {
        TDateTime today = TDateTime::CurrentDateTime();
        reg->WriteDateTime("Creation\ Date", today);
    }

    if(!reg->KeyExists("Licenced\ To"))
    {
        reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
    }

    if(!reg->KeyExists("App\ Location"))
    {
        reg->WriteExpandString("App\ Location",
                               "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
    }

    if(!reg->KeyExists("Projects\ Location"))
    {
        reg->WriteExpandString("Projects\ Location",
                               "%USERPROFILE%\\MyApplication\\Projects\\");
    }

    reg->CloseKey();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
{
    //deleting the example registry entries
    TRegistry* reg = new TRegistry(KEY_WRITE);
    reg->RootKey = HKEY_LOCAL_MACHINE;

    reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
    reg->DeleteKey("Software\\MyCompanyName");

    reg->CloseKey();
}
//---------------------------------------------------------------------------
```

Delphi Examples:

```
{
This example shows how to use the TRegistry class in order to
find, insert and delete Keys and Items into the Windows Registry.
This example uses two buttons: InsertToRegBtn and DeleteFromRegBtn,
on a form, for inserting and deleting the values.
}
procedure TForm3.InsertToRegBtnClick(Sender: TObject);
var
  reg        : TRegistry;
  openResult : Boolean;
  today      : TDateTime;
begin
  reg := TRegistry.Create(KEY_READ);
  reg.RootKey := HKEY_LOCAL_MACHINE;

  if (not reg.KeyExists('Software\\MyCompanyName\\MyApplication\\')) then
    begin
      MessageDlg('Key not found! Created now.',
                          mtInformation, mbOKCancel, 0);
    end;
  reg.Access := KEY_WRITE;
```

```
    openResult := reg.OpenKey('Software\\MyCompanyName\\MyApplication\\',True);

    if not openResult = True then
      begin
        MessageDlg('Unable to create key! Exiting.',
                   mtError, mbOKCancel, 0);
        Exit();
      end;

    {checking if the values exist and inserting when neccesary}

    if not reg.KeyExists('Creation\ Date') then
      begin
        today := Now;
          reg.WriteDateTime('Creation\ Date', today);
      end;

    if not reg.KeyExists('Licenced\ To') then
      begin
          reg.WriteString('Licenced\ To', 'MySurname\ MyFirstName');
      end;

    if not reg.KeyExists('App\ Location') then
      begin
          reg.WriteExpandString('App\ Location',
                                '%PROGRAMFILES%\\MyCompanyName\\MyApplication\\');
      end;

    if not reg.KeyExists('Projects\ Location') then
      begin
          reg.WriteExpandString('Projects\ Location',
                                '%USERPROFILE%\\MyApplication\\Projects\\');
      end;

    reg.CloseKey();

  end;

  procedure TForm3.DeleteFromRegBtnClick(Sender: TObject);
  var
    reg : TRegistry;
  begin
    reg := TRegistry.Create(KEY_WRITE);
    reg.RootKey := HKEY_LOCAL_MACHINE;

    reg.DeleteKey('Software\\MyCompanyName\\MyApplication');
    reg.DeleteKey('Software\\MyCompanyName');

    reg.CloseKey();
  end;
```

See Also

CurrentKey (see page 3296)

LoadKey (see page 3268)

RegistryConnect (see page 3277)

59.1.4 TRegistryIniFile Class

TRegistryIniFile is a wrapper for the Windows system registry.

Class Hierarchy

Pascal

```
TRegistryIniFile = class(TCustomIniFile);
```

C++

```
class TRegistryIniFile : public TCustomIniFile;
```

File

Registry

Description

TRegistryIniFile presents a simple interface to the system registry and hides the need to know about the underlying structure of the registry. TRegistryIniFile enables handling the Windows system registry as if it were a Windows 3.x INI file. Instead of processing an INI file, however, TRegistryIniFile reads from and writes to the system registry. Because TRegistryIniFile, like TIniFile (see page 752), descends from TCustomIniFile (see page 716), you can use it in common code that writes to both INI files and the registry.

In addition, application developers that are switching from the use of INI files to the system registry can use this object to migrate their application with a minimum of coding changes. By finding all references to TIniFile (see page 752) in an application, replacing them with TRegistryIniFile, and recompiling the application, a developer can update an application to use the system registry instead of INI files without having to code any new logic into the application.

Note: Information contained in existing users' INI files is not migrated into the system registry using the method described above. A one-time operation to explicitly copy the INI file data, through TRegistryIniFile, must be used to do this.

TRegistryIniFile adapts the methods inherited from TCustomIniFile (see page 716) to operate on Registry (see page 3235) entries rather than INI file keys. It reinterprets the FileName property as a subkey under the system registry's root key (HKEY_CURRENT_USER by default). What corresponds to a section in an INI file is treated as a key in the system registry, and what corresponds to data entries under a section in an INI file are treated as data values under a key in the system registry.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}
```

```
void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
```

```
      {
      Read all saved values from the last session. The section name
      is the name of the form. Also use form's properties as defaults
      }
      Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
      Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
      Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
      Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
      Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

      { Load last window state }
      case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
        true : WindowState := wsMaximized;
        false: WindowState := wsNormal;
      end;

    finally
      SettingsFile.Free;
    end;
  end;

  procedure TForm2.btStoreClick(Sender: TObject);
  var
    SettingsFile: TCustomIniFile;
  begin
    { Open an instance }
    SettingsFile := OpenIniFileInstance();

    {
    Store current form properties to be used in later sessions.
    }
    try
      SettingsFile.WriteInteger (Name, 'Top', Top);
      SettingsFile.WriteInteger (Name, 'Left', Left);
      SettingsFile.WriteInteger (Name, 'Width', Width);
      SettingsFile.WriteInteger (Name, 'Height', Height);
      SettingsFile.WriteString  (Name, 'Caption', Caption);
      SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
      SettingsFile.WriteDateTime(Name, 'LastRun', Now);
    finally
      SettingsFile.Free;
    end;

  end;

  function TForm2.OpenIniFileInstance: TCustomIniFile;
  begin
    {
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    }

    case RadioGroup1.ItemIndex of
      0:
        begin
          { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
          Result := TRegistryIniFile.Create('Software\' + Application.Title);
        end;
      1:
        begin
          { Ini file mode selected }
          Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
      2:
        begin
          { Memory based Ini file mode selected }
```

```
        Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
  end;
end;
```

See Also

TIniFile (see page 752)

TMemIniFile (see page 765)

TCustomIniFile (see page 716)

TRegIniFile (see page 3237)

59.1.4.1 TRegistryIniFile Members

The following tables list the members exposed by TRegistryIniFile.

Public Methods

	Name	Description
	Create (see page 3304)	This is the overview for the Create constructor overload.
	DeleteKey (see page 3305)	Erases the data value associated with an INI file entry.
	Destroy (see page 3304)	Disposes of an object instance.
	EraseSection (see page 3306)	Erases an entire section of an INI file.
	ReadBinaryStream (see page 3306)	Retrieves a binary value from the registry.
	ReadBool (see page 723)	Retrieves a Boolean value from an ini file.
	ReadDate (see page 3307)	Retrieves a date value from an INI file.
	ReadDateTime (see page 3307)	Retrieves a date-time value from an INI file.
	ReadFloat (see page 3308)	Retrieves a float value from an INI file.
	ReadInteger (see page 3308)	Retrieves an integer value from an INI file.
	ReadSection (see page 3311)	Reads all the key names from a specified section of an INI file into a string list.
	ReadSections (see page 731)	This is the overview for the ReadSections method overload.
	ReadSections (see page 3312)	This is the overview for the ReadSections method overload.
	ReadSectionValues (see page 3313)	Retrieves the data-values associated with a specified key, and stores the retrieved values in a string object.
	ReadString (see page 3313)	Retrieves a string value from an INI file.
	ReadTime (see page 3316)	Retrieves a time value from an INI file.
	SectionExists (see page 736)	Indicates whether a section exists in the ini file.
	UpdateFile (see page 3317)	Flushes buffered INI file data to disk.
	ValueExists (see page 736)	Indicates whether a key exists in the ini file.
	WriteBinaryStream (see page 3317)	Writes a binary value to the registry.
	WriteBool (see page 737)	Writes a Boolean value to an ini file.
	WriteDate (see page 3317)	Writes a date value to an INI file.
	WriteDateTime (see page 3318)	Writes a date-time value to an INI file.
	WriteFloat (see page 3319)	Writes a float value to an INI file.
	WriteInteger (see page 3319)	Writes an integer value to an INI file.
	WriteString (see page 3322)	Writes a string value to an INI file.
	WriteTime (see page 3325)	Writes a time value to an INI file.

Public Properties

	Name	Description
	FileName (see page 749)	Contains the name of the ini file from which to read and to which to write information.
	RegIniFile (see page 3326)	Contains and internal instance of a TRegIniFile (see page 3237) object.

59.1.4.2 Create Constructor

This is the overview for the Create constructor overload.

Overload List

	Name	Description
	TRegistryIniFile.Create (string) (see page 3304)	Creates a TRegistryIniFile (see page 3299) object for an application.
	TRegistryIniFile.Create (string, LongWord) (see page 3304)	Creates a TRegistryIniFile (see page 3299) object for an application.

59.1.4.2.1 TRegistryIniFile.Create Constructor (string)

Creates a TRegistryIniFile (see page 3299) object for an application.

Pascal

```
constructor Create(const FileName: string); overload;
constructor Create(const FileName: string; AAccess: LongWord); overload;
```

C++

```
__fastcall TRegistryIniFile(const UnicodeString FileName);
__fastcall TRegistryIniFile(const UnicodeString FileName, LongWord AAccess);
```

Description

Create constructs a TRegistryIniFile (see page 3299) and creates an internal TRegIniFile (see page 3237) object used to access the system registry. TRegistryIniFile (see page 3299) uses this TRegIniFile (see page 3237) object as a helper object to provide much of its functionality.

FileName is used as the FileName property for the internal TRegIniFile (see page 3237) object in the RegIniFile (see page 3326) property.

AAccess initializes the Access property of the internal TRegIniFile (see page 3237) object.

See Also

TRegIniFile (see page 3237)

RegIniFile (see page 3326)

System::TObject::Create

Access (see page 3293)

59.1.4.3 TRegistryIniFile.Destroy Destructor

Disposes of an object instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRegistryIniFile();
```

Description

Do not call System::TObject::Destroy directly. Call System::TObject::Free instead. Free verifies that the object reference is not nil before calling System::TObject::Destroy.

The System::TObject::Destroy method defined by System::TObject deallocates memory. Descendant objects usually define a destructor that is customized for that particular kind of object.

When declaring a System::TObject::Destroy method in a descendant, always add the override directive to the declaration and call the inherited System::TObject::Destroy as the last statement in the overriding method. Since System::TObject::Destroy is a virtual method, overriding it ensures that the proper inherited behavior occurs.

Note: If an exception escapes from the constructor, the destructor is called to destroy the partially constructed object instance that failed to initialize completely. Therefore, destructors should check that allocated resources, such as handles, were actually allocated before trying to release them, since their value might be zero.

System::TObject::Destroy should be implemented so that it calls System::TObject::Free on all subobjects created within the object's constructor (that is, allocated by the constructor). Unlike System::TObject::Destroy, System::TObject::Free provides a safeguard when destroying objects that are nil.

See Also

System::TObject::Free

59.1.4.4 TRegistryIniFile Methods

The methods of the TRegistryIniFile class are listed here.

Public Methods

	Name	Description
	DeleteKey (see page 3305)	Erases the data value associated with an INI file entry.
	EraseSection (see page 3306)	Erases an entire section of an INI file.
	ReadBinaryStream (see page 3306)	Retrieves a binary value from the registry.
	ReadDate (see page 3307)	Retrieves a date value from an INI file.
	ReadDateTime (see page 3307)	Retrieves a date-time value from an INI file.
	ReadFloat (see page 3308)	Retrieves a float value from an INI file.
	ReadInteger (see page 3308)	Retrieves an integer value from an INI file.
	ReadSection (see page 3311)	Reads all the key names from a specified section of an INI file into a string list.
	ReadSections (see page 3312)	This is the overview for the ReadSections method overload.
	ReadSectionValues (see page 3313)	Retrieves the data-values associated with a specified key, and stores the retrieved values in a string object.
	ReadString (see page 3313)	Retrieves a string value from an INI file.
	ReadTime (see page 3316)	Retrieves a time value from an INI file.
	UpdateFile (see page 3317)	Flushes buffered INI file data to disk.
	WriteBinaryStream (see page 3317)	Writes a binary value to the registry.
	WriteDate (see page 3317)	Writes a date value to an INI file.
	WriteDateTime (see page 3318)	Writes a date-time value to an INI file.
	WriteFloat (see page 3319)	Writes a float value to an INI file.
	WriteInteger (see page 3319)	Writes an integer value to an INI file.
	WriteString (see page 3322)	Writes a string value to an INI file.
	WriteTime (see page 3325)	Writes a time value to an INI file.

59.1.4.4.1 TRegistryIniFile.DeleteKey Method

Erases the data value associated with an INI file entry.

Pascal

```
procedure DeleteKey(const Section: string; const Ident: string); override;
```

C++

```
virtual __fastcall DeleteKey(const UnicodeString Section, const UnicodeString Ident);
```

Description

Call DeleteKey to erase a data value associated with an INI file entry. Section is string containing the name of an INI file section, and Ident is a string containing the name of the key from for which to set a nil (Delphi) or NULL (C++) value.

Note: Attempting to erase a data value in a nonexistent section or attempting to erase data from a nonexistent key are not errors. In these cases, DeleteKey creates the section and key and sets its initial value to an empty value.

See Also

EraseSection (see page 3306)

ValueExists (see page 736)

59.1.4.4.2 TRegistryIniFile.EraseSection Method

Erases an entire section of an INI file.

Pascal

```
procedure EraseSection(const Section: string); override;
```

C++

```
virtual __fastcall EraseSection(const UnicodeString Section);
```

Description

Call EraseSection to remove a section, all its keys, and their data values from an INI file. Section identifies the INI file section to remove. If a section cannot be removed, an exception is raised.

See Also

SectionExists (see page 736)

DeleteKey (see page 3305)

59.1.4.4.3 TRegistryIniFile.ReadBinaryStream Method

Retrieves a binary value from the registry.

Pascal

```
function ReadBinaryStream(const Section: string; const Name: string; Value: TStream): Integer;
override;
```

C++

```
virtual __fastcall int ReadBinaryStream(const UnicodeString Section, const UnicodeString Name,
TStream Value);
```

Description

Call ReadBinaryStream to read a binary value into a specified stream. After reading the binary value from the INI file into the current position of the stream, ReadBinaryStream returns the number of bytes written to the stream. The stream is left positioned immediately after the new binary value.

Section is the key for the section that contains the binary data.

Name is the subkey under which the binary data is stored.

Value is a stream to which the binary value is written.

See Also

ReadString (see page 3313)

WriteBinaryStream (see page 3317)

59.1.4.4.4 TRegistryIniFile.ReadDate Method

Retrieves a date value from an INI file.

Pascal

```
function ReadDate(const Section: string; const Name: string; Default: TDateTime): TDateTime;
override;
```

C++

```
virtual __fastcall TDateTime ReadDate(const UnicodeString Section, const UnicodeString Name,
TDateTime Default);
```

Description

Call ReadDate to read a date value from an INI file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the date value. Default is the date value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

See Also

ReadBool (see page 723)

ReadDateTime (see page 3307)

ReadFloat (see page 3308)

ReadInteger (see page 3308)

ReadString (see page 3313)

ReadTime (see page 3316)

59.1.4.4.5 TRegistryIniFile.ReadDateTime Method

Retrieves a date-time value from an INI file.

Pascal

```
function ReadDateTime(const Section: string; const Name: string; Default: TDateTime):
TDateTime; override;
```

C++

```
virtual __fastcall TDateTime ReadDateTime(const UnicodeString Section, const UnicodeString
Name, TDateTime Default);
```

Description

Call ReadDateTime to read a date-time value from an INI file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the date-time value. Default is the date-time value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

See Also

ReadBool (see page 723)

ReadDate (see page 3307)

ReadFloat (see page 3308)

ReadInteger (see page 3308)

ReadString (see page 3313)

ReadTime (see page 3316)

59.1.4.4.6 TRegistryIniFile.ReadFloat Method

Retrieves a float value from an INI file.

Pascal

```
function ReadFloat(const Section: string; const Name: string; Default: Double): Double;
override;
```

C++

```
virtual __fastcall Double ReadFloat(const UnicodeString Section, const UnicodeString Name,
Double Default);
```

Description

Call ReadFloat to read a float value from an INI file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the float value. Default is the float value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

See Also

ReadBool (see page 723)

ReadDate (see page 3307)

ReadDateTime (see page 3307)

ReadInteger (see page 3308)

ReadString (see page 3313)

ReadTime (see page 3316)

59.1.4.4.7 TRegistryIniFile.ReadInteger Method

Retrieves an integer value from an INI file.

Pascal

```
function ReadInteger(const Section: string; const Ident: string; Default: Longint): Longint;
override;
```

C++

```
virtual __fastcall Longint ReadInteger(const UnicodeString Section, const UnicodeString Ident,
Longint Default);
```

Description

Call ReadInteger to read an integer value from an INI file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the value. Default is the integer value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
```

```
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
```

```
    SettingsFile: TCustomIniFile;
  begin
    { Open an instance }
    SettingsFile := OpenIniFileInstance();

    {
    Store current form properties to be used in later sessions.
    }
    try
      SettingsFile.WriteInteger (Name, 'Top', Top);
      SettingsFile.WriteInteger (Name, 'Left', Left);
      SettingsFile.WriteInteger (Name, 'Width', Width);
      SettingsFile.WriteInteger (Name, 'Height', Height);
      SettingsFile.WriteString  (Name, 'Caption', Caption);
      SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
      SettingsFile.WriteDateTime(Name, 'LastRun', Now);
    finally
      SettingsFile.Free;
    end;

  end;

  function TForm2.OpenIniFileInstance: TCustomIniFile;
  begin
    {
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    }

    case RadioGroup1.ItemIndex of
      0:
        begin
          { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
          Result := TRegistryIniFile.Create('Software\' + Application.Title);
        end;
      1:
        begin
          { Ini file mode selected }
          Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
      2:
        begin
          { Memory based Ini file mode selected }
          Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
    end;
  end;
```

See Also

ReadBool (see page 723)

ReadDate (see page 3307)

ReadDateTime (see page 3307)

ReadFloat (see page 3308)

ReadString (see page 3313)

ReadTime (see page 3316)

59.1.4.4.8 TRegistryIniFile.ReadSection Method

Reads all the key names from a specified section of an INI file into a string list.

Pascal

```
procedure ReadSection(const Section: string; Strings: TStrings); override;
```

C++

```
virtual __fastcall ReadSection(const UnicodeString Section, TStrings Strings);
```

Description

Call ReadSection to retrieve the names of all keys within a specified section of an INI file into a string list object.

Section identifies the section from which to retrieve a list of key names. Strings specifies the string object to hold the retrieved names. Strings can point to a TStrings object such as a string list, or to a component property, such as Items for a TListBox component, that is a TStrings object itself.

See Also

ReadSections (see page 3312)

ReadSectionValues (see page 3313)

SectionExists (see page 736)

59.1.4.4.9 ReadSections Method

This is the overview for the ReadSections method overload.

Overload List

	Name	Description
	TRegistryIniFile.ReadSections (string, TStrings) (see page 3312)	Reads the names of all sections in an INI file into a string list.
	TRegistryIniFile.ReadSections (TStrings) (see page 3312)	Reads the names of all sections in an INI file into a string list.

59.1.4.4.9.1 TRegistryIniFile.ReadSections Method (TStrings)

Reads the names of all sections in an INI file into a string list.

Pascal

```
procedure ReadSections(Strings: TStrings); override; overload;
procedure ReadSections(const Section: string; Strings: TStrings); override; overload;
```

C++

```
virtual __fastcall ReadSections(TStrings Strings);
virtual __fastcall ReadSections(const UnicodeString Section, TStrings Strings);
```

Description

Call ReadSections to retrieve the names of all sections in an INI file into a string list object.

Strings specifies the string object to hold the retrieved names. Strings can point to a TStrings object, or to a component property, such as Items for a TListBox component, that is a TStrings object itself.

See Also

ReadSection (see page 3311)

ReadSectionValues (see page 3313)

59.1.4.4.10 TRegistryIniFile.ReadSectionValues Method

Retrieves the data-values associated with a specified key, and stores the retrieved values in a string object.

Pascal

```
procedure ReadSectionValues(const Section: string; Strings: TStrings); override;
```

C++

```
virtual __fastcall ReadSectionValues(const UnicodeString Section, TStrings Strings);
```

Description

Call ReadSectionValues to read all data values associated with a specified key into a string object.

Section is the name of the key from which to retrieve data-values.

Strings specifies the string list object to hold the retrieved values. Strings can point to a TStrings object, or to a component property, such as Items for a TListBox component, that is a TStrings object itself. Use the Values property of the string object to access a specific string in the object.

See Also

ReadSection (see page 3311)

ReadSections (see page 3312)

SectionExists (see page 736)

59.1.4.4.11 TRegistryIniFile.ReadString Method

Retrieves a string value from an INI file.

Pascal

```
function ReadString(const Section: string; const Ident: string; const Default: string):
string; override;
```

C++

```
virtual __fastcall UnicodeString ReadString(const UnicodeString Section, const UnicodeString
Ident, const UnicodeString Default);
```

Description

Call ReadString to read a string value from an INI file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the value. Default is the string value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();
```

```
    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
```

```
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
    SettingsFile.WriteInteger (Name, 'Height', Height);
    SettingsFile.WriteString  (Name, 'Caption', Caption);
    SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
    SettingsFile.WriteDateTime(Name, 'LastRun', Now);
  finally
    SettingsFile.Free;
  end;

end;

function TForm2.OpenIniFileInstance: TCustomIniFile;
begin
  {
  Open/create a new INI file that has the same name as our executable
```

```
    only with the INI extension.
    }

    case RadioGroup1.ItemIndex of
      0:
        begin
          { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
          Result := TRegistryIniFile.Create('Software\' + Application.Title);
        end;
      1:
        begin
          { Ini file mode selected }
          Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
      2:
        begin
          { Memory based Ini file mode selected }
          Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
        end;
    end;
  end;
```

See Also

ReadBool (see page 723)

ReadDate (see page 3307)

ReadDateTime (see page 3307)

ReadFloat (see page 3308)

ReadInteger (see page 3308)

ReadTime (see page 3316)

59.1.4.4.12 TRegistryIniFile.ReadTime Method

Retrieves a time value from an INI file.

Pascal

```
function ReadTime(const Section: string; const Name: string; Default: TDateTime): TDateTime;
override;
```

C++

```
virtual __fastcall TDateTime ReadTime(const UnicodeString Section, const UnicodeString Name,
TDateTime Default);
```

Description

Call ReadTime to read a time value from an INI file. Section identifies the section in the file that contains the desired key. Ident is the name of the key from which to retrieve the time value. Default is the time value to return if the:

Section does not exist.

Key does not exist.

Data value for the key is not assigned.

See Also

ReadBool (see page 723)

ReadDate (see page 3307)

ReadDateTime (see page 3307)

ReadFloat (see page 3308)

ReadInteger (see page 3308)

ReadString (see page 3313)

59.1.4.4.13 TRegistryIniFile.UpdateFile Method

Flushes buffered INI file data to disk.

Pascal

```
procedure UpdateFile; override;
```

C++

```
virtual __fastcall UpdateFile();
```

Description

Call UpdateFile to flush buffered reads from and writes to the INI file to disk. UpdateFile is useful under Windows95, but has no effect under Windows NT as NT does not buffer INI file reads and writes.

See Also

FileName (see page 749)

59.1.4.4.14 TRegistryIniFile.WriteBinaryStream Method

Writes a binary value to the registry.

Pascal

```
procedure WriteBinaryStream(const Section: string; const Name: string; Value: TStream);
override;
```

C++

```
virtual __fastcall WriteBinaryStream(const UnicodeString Section, const UnicodeString Name,
TStream Value);
```

Description

Call WriteBinaryStream to write a binary value from a specified stream to a specified key in the registry. WriteBinaryStream writes from the current position in the stream to the end.

Section is the key for the section that contains the subkey where the binary data should be written.

Name is the subkey under which the binary data is stored.

Value is a stream from which the binary value can be read.

Note: Attempting to write a value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteBinaryStream creates the section and key and sets its initial value to the stream's data.

See Also

WriteString (see page 3322)

ReadBinaryStream (see page 3306)

59.1.4.4.15 TRegistryIniFile.WriteDate Method

Writes a date value to an INI file.

Pascal

```
procedure WriteDate(const Section: string; const Name: string; Value: TDateTime); override;
```

C++

```
virtual __fastcall WriteDate(const UnicodeString Section, const UnicodeString Name, TDateTime
Value);
```

Description

Call WriteDate to write a date value to an INI file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the date value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteDate creates the section and key and sets its initial value to Value.

See Also

WriteBool (see page 737)

WriteDateTime (see page 3318)

WriteFloat (see page 3319)

WriteInteger (see page 3319)

WriteString (see page 3322)

WriteTime (see page 3325)

59.1.4.4.16 TRegistryIniFile.WriteDateTime Method

Writes a date-time value to an INI file.

Pascal

```
procedure WriteDateTime(const Section: string; const Name: string; Value: TDateTime); override;
```

C++

```
virtual __fastcall WriteDateTime(const UnicodeString Section, const UnicodeString Name,
TDateTime Value);
```

Description

Call WriteDateTime to write a date-time value to an INI file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the date-time value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteDateTime creates the section and key and sets its initial value to Value.

See Also

WriteBool (see page 737)

WriteDate (see page 3317)

WriteFloat (see page 3319)

WriteInteger (see page 3319)

WriteString (see page 3322)

WriteTime (see page 3325)

59.1.4.4.17 TRegistryIniFile.WriteFloat Method

Writes a float value to an INI file.

Pascal

```
procedure WriteFloat(const Section: string; const Name: string; Value: Double); override;
```

C++

```
virtual __fastcall WriteFloat(const UnicodeString Section, const UnicodeString Name, Double
Value);
```

Description

Call WriteFloat to write a float value to an INI file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the float value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteFloat creates the section and key and sets its initial value to Value.

See Also

WriteBool (see page 737)

WriteDate (see page 3317)

WriteDateTime (see page 3318)

WriteInteger (see page 3319)

WriteString (see page 3322)

WriteTime (see page 3325)

59.1.4.4.18 TRegistryIniFile.WriteInteger Method

Writes an integer value to an INI file.

Pascal

```
procedure WriteInteger(const Section: string; const Ident: string; Value: Longint); override;
```

C++

```
virtual __fastcall WriteInteger(const UnicodeString Section, const UnicodeString Ident,
Longint Value);
```

Description

Call WriteInteger to write an integer value to an INI file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the integer value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteInteger creates the section and key and sets its initial value to Value.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
```

```
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
```

```
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
      }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
    SettingsFile.WriteInteger (Name, 'Height', Height);
    SettingsFile.WriteString  (Name, 'Caption', Caption);
    SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
    SettingsFile.WriteDateTime(Name, 'LastRun', Now);
  finally
    SettingsFile.Free;
  end;

end;

function TForm2.OpenIniFileInstance: TCustomIniFile;
```

```
begin
  {
  Open/create a new INI file that has the same name as our executable
  only with the INI extension.
  }

  case RadioGroup1.ItemIndex of
    0:
      begin
        { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
        Result := TRegistryIniFile.Create('Software\' + Application.Title);
      end;
    1:
      begin
        { Ini file mode selected }
        Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
    2:
      begin
        { Memory based Ini file mode selected }
        Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
  end;
end;
```

See Also

WriteBool (see page 737)

WriteDate (see page 3317)

WriteDateTime (see page 3318)

WriteFloat (see page 3319)

WriteString (see page 3322)

WriteTime (see page 3325)

59.1.4.4.19 TRegistryIniFile.WriteString Method

Writes a string value to an INI file.

Pascal

```
procedure WriteString(const Section: string; const Ident: string; const Value: string);
override;
```

C++

```
virtual __fastcall WriteString(const UnicodeString Section, const UnicodeString Ident, const
UnicodeString Value);
```

Description

Call WriteString to write a string value to an INI file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the string value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteString creates the section and key and sets its initial value to Value.

C++ Examples:

```
/*
This example demonstrates the usage of Ini files in order to store
```

```
and load form configuration between sessions. Example assumes
a RadioGroup and two buttons are present on the main form.
*/
void __fastcall TForm2::btStoreClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    // Store current form properties to be used in later sessions.
    try
    {
        SettingsFile->WriteInteger (Name, "Top", Top);
        SettingsFile->WriteInteger (Name, "Left", Left);
        SettingsFile->WriteInteger (Name, "Width", Width);
        SettingsFile->WriteInteger (Name, "Height", Height);
        SettingsFile->WriteString  (Name, "Caption", Caption);
        SettingsFile->WriteBool    (Name, "InitMax", WindowState == wsMaximized );
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

void __fastcall TForm2::btLoadClick(TObject *Sender)
{
    /* Open an instance */
    TCustomIniFile* SettingsFile = OpenIniFileInstance();

    try
    {
        /*
        Read all saved values from the last session. The section name
        is the name of the form. Also use form's properties as defaults
        */
        Top     = SettingsFile->ReadInteger(Name, "Top", Top );
        Left    = SettingsFile->ReadInteger(Name, "Left", Left );
        Width   = SettingsFile->ReadInteger(Name, "Width", Width );
        Height  = SettingsFile->ReadInteger(Name, "Height", Height );
        Caption = SettingsFile->ReadString (Name, "Caption", Caption);

        // Load last window state
        if (SettingsFile->ReadBool(Name, "InitMax", WindowState == wsMaximized))
            WindowState = wsMaximized;
        else
            WindowState = wsNormal;
    }
    catch(Exception* e)
    {
    }

    delete SettingsFile;
}

TCustomIniFile* __fastcall TForm2::OpenIniFileInstance()
{
    /*
    Open/create a new INI file that has the same name as our executable
    only with the INI extension.
    */

    switch (RadioGroup1->ItemIndex)
    {
        case 0:
            /* Registry mode selected: in HKEY_CURRENT_USER\Software\... */
```

```
            return new TRegistryIniFile(String("Software\\") + Application->Title);
        case 1:
            /* Ini file mode selected */
            return new TIniFile(ChangeFileExt(Application->ExeName, ".INI"));
        case 2:
            /* Memory based Ini file mode selected */
            return new TMemIniFile(ChangeFileExt(Application->ExeName, ".INI"));
    }
}
```

Delphi Examples:

```
{
This example demonstrates the use of Ini files in order to store
and load form configurations between sessions. This example requires
a RadioGroup and two buttons.
}
procedure TForm2.btLoadClick(Sender: TObject);
var
  SettingsFile : TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  try
    {
    Read all saved values from the last session. The section name
    is the name of the form. Also use form's properties as defaults
    }
    Top     := SettingsFile.ReadInteger(Name, 'Top', Top );
    Left    := SettingsFile.ReadInteger(Name, 'Left', Left );
    Width   := SettingsFile.ReadInteger(Name, 'Width', Width );
    Height  := SettingsFile.ReadInteger(Name, 'Height', Height );
    Caption := SettingsFile.ReadString (Name, 'Caption', Caption);

    { Load last window state }
    case SettingsFile.ReadBool(Name, 'InitMax', WindowState = wsMaximized) of
      true : WindowState := wsMaximized;
      false: WindowState := wsNormal;
    end;

  finally
    SettingsFile.Free;
  end;
end;

procedure TForm2.btStoreClick(Sender: TObject);
var
  SettingsFile: TCustomIniFile;
begin
  { Open an instance }
  SettingsFile := OpenIniFileInstance();

  {
  Store current form properties to be used in later sessions.
  }
  try
    SettingsFile.WriteInteger (Name, 'Top', Top);
    SettingsFile.WriteInteger (Name, 'Left', Left);
    SettingsFile.WriteInteger (Name, 'Width', Width);
    SettingsFile.WriteInteger (Name, 'Height', Height);
    SettingsFile.WriteString  (Name, 'Caption', Caption);
    SettingsFile.WriteBool    (Name, 'InitMax', WindowState = wsMaximized );
    SettingsFile.WriteDateTime(Name, 'LastRun', Now);
  finally
    SettingsFile.Free;
  end;
```

```
end;

function TForm2.OpenIniFileInstance: TCustomIniFile;
begin
  {
  Open/create a new INI file that has the same name as our executable
  only with the INI extension.
  }

  case RadioGroup1.ItemIndex of
    0:
      begin
        { Registry mode selected: in HKEY_CURRENT_USER\Software\... }
        Result := TRegistryIniFile.Create('Software\' + Application.Title);
      end;
    1:
      begin
        { Ini file mode selected }
        Result := TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
    2:
      begin
        { Memory based Ini file mode selected }
        Result := TMemIniFile.Create(ChangeFileExt(Application.ExeName, '.INI'));
      end;
  end;
end;
```

See Also

WriteBool (see page 737)

WriteDate (see page 3317)

WriteDateTime (see page 3318)

WriteFloat (see page 3319)

WriteInteger (see page 3319)

WriteTime (see page 3325)

59.1.4.4.20 TRegistryIniFile.WriteTime Method

Writes a time value to an INI file.

Pascal

```
procedure WriteTime(const Section: string; const Name: string; Value: TDateTime); override;
```

C++

```
virtual __fastcall WriteTime(const UnicodeString Section, const UnicodeString Name, TDateTime
Value);
```

Description

Call WriteTime to write a time value to an INI file. Section identifies the section in the file that contain the key to which to write. Ident is the name of the key for which to set a value. Value is the time value to write.

Note: Attempting to write a data value to a nonexistent section or attempting to write data to a nonexistent key are not errors. In these cases, WriteTime creates the section and key and sets its initial value to Value.

See Also

WriteBool (see page 737)

WriteDate (see page 3317)

WriteDateTime (see page 3318)

WriteFloat (see page 3319)

WriteInteger (see page 3319)

WriteString (see page 3322)

59.1.4.5 TRegistryIniFile Properties

The properties of the TRegistryIniFile class are listed here.

Public Properties

	Name	Description
	RegIniFile (see page 3326)	Contains and internal instance of a TRegIniFile (see page 3237) object.

59.1.4.5.1 TRegistryIniFile.RegIniFile Property

Contains and internal instance of a TRegIniFile (see page 3237) object.

Pascal

```
property RegIniFile: TRegIniFile;
```

C++

```
__property TRegIniFile RegIniFile;
```

Description

RegIniFile is used internally by the TRegistryIniFile (see page 3299) object as the gateway connection between TRegistryIniFile (see page 3299) methods and the system registry.

See Also

Create (see page 3304)

TRegIniFile (see page 3237)

59.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TRegDataType (see page 3327)	TRegDataType is a type used by the GetDataInfo method of TRegistery.

Records

	Name	Description
	TRegDataInfo (see page 3327)	TRegDataInfo is a type used by the GetDataInfo method of TRegistery.
	TRegKeyInfo (see page 3328)	TRegKeyInfo is a type used by the GetRegKeyInfo method of TRegistery.

59.2.1 Registry.TRegDataInfo Record

TRegDataInfo is a type used by the GetDataInfo method of TRegistery.

Pascal

```
TRegDataInfo = record
  RegData: TRegDataType;
  DataSize: Integer;
end;
```

C++

```
struct TRegDataInfo {
  TRegDataType RegData;
  int DataSize;
};
```

File

Registry

Description

TRegDataInfo is a type used by the GetDataInfo method of TRegistery. Records of type TRegDataInfo contain two fields: a field called RegData, of type TRegDataType (see page 3327), that specifies the data type of the data value; and a field called DataSize that specifies the size, in bytes, of the data value.

59.2.2 Registry.TRegDataType Enumeration

TRegDataType is a type used by the GetDataInfo method of TRegistery.

Pascal

```
TRegDataType = (
  rdUnknown,
  rdString,
  rdExpandString,
  rdInteger,
```

```
    rdBinary
  );
```

C++

```
  enum TRegDataType {
    rdUnknown,
    rdString,
    rdExpandString,
    rdInteger,
    rdBinary
  };
```

File

Registry

Description

TRegDataType is a type used by the GetDataInfo method of TRegistery.It can have the following values:

Value	Meaning
rdUnknown	Data value type is of no defined value type.
rdString	Data value type is a null-terminated string. It is a Unicode or ANSI string depending on whether the application calls the Unicode or ANSI string functions.
rdExpandString	Data value type is a null-terminated string that contains unexpanded references to environment variables (for example, "%PATH%"). It is a Unicode or ANSI string depending on whether the application calls the Unicode or ANSI string functions.
rdInteger	Data value type is a32-bit number.
rdBinary	Data value type is binary data, in any form.

59.2.3 Registry.TRegKeyInfo Record

TRegKeyInfo is a type used by the GetRegKeyInfo method of TRegistery.

Pascal

```
  TRegKeyInfo = record
    NumSubKeys: Integer;
    MaxSubKeyLen: Integer;
    NumValues: Integer;
    MaxValueLen: Integer;
    MaxDataLen: Integer;
    FileTime: TFileTime;
  end;
```

C++

```
  struct TRegKeyInfo {
    int NumSubKeys;
    int MaxSubKeyLen;
    int NumValues;
    int MaxValueLen;
    int MaxDataLen;
    TFileTime FileTime;
  };
```

File

Registry

Description

TRegKeyInfo is a type used by the GetRegKeyInfo method of TRegistery to return information about the current key:

Note: Number of subkeys

Note: Longest subkey name length, in characters

Note: Number of data values

Note: Longest data-value name length

Note: Longest data-value length

Note: Time of last write to the key

60 ReqFiles Namespace

This is namespace ReqFiles.

Topics

Name	Description
Classes (see page 3332)	The following table lists classes in this documentation.

Classes

	Name	Description
	TWebRequestFile (see page 3332)	TWebRequestFile describes the content of a file extracted from an HTTP request.
	TWebRequestFiles (see page 3336)	TWebRequestFiles stores a collection of TWebAbstractRequestFile objects.
	TWebRequestFileStream (see page 3339)	This is class ReqFiles.TWebRequestFileStream.

60.1 Classes

The following table lists classes in this documentation.

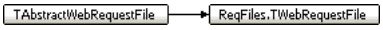
Classes

	Name	Description
	TWebRequestFile (see page 3332)	TWebRequestFile describes the content of a file extracted from an HTTP request.
	TWebRequestFiles (see page 3336)	TWebRequestFiles stores a collection of TWebAbstractRequestFile objects.
	TWebRequestFileStream (see page 3339)	This is class ReqFiles.TWebRequestFileStream.

60.1.1 TWebRequestFile Class

TWebRequestFile describes the content of a file extracted from an HTTP request.

Class Hierarchy



Pascal

```
TWebRequestFile = class(TAbstractWebRequestFile);
```

C++

```
class TWebRequestFile : public TAbstractWebRequestFile;
```

File

ReqFiles

Description

TWebRequestFile describes and provides access to an uploaded file from an HTTP request. The following properties support this access:

FileName, which gives the name of the uploaded file.

FieldName, which gives the name of the adapter field whose value is the uploaded file.

ContentType, which describes how the file data is encoded.

Stream, which is a stream object from which you can read the file content.

Each TWebRequestFile object describes a single uploaded file. A collection of all the uploaded files from the current request is available as the Files property of TWebRequest. In addition, adapter fields that work with uploaded files surface those files using a TUpdateFileList object or IActionFieldValues interface.

See Also

TWebRequestFiles (see page 3336)

IActionFieldValues

TUpdateFileList

TAdapterFileField

60.1.1.1 TWebRequestFile Members

The following tables list the members exposed by TWebRequestFile.

Protected Methods

	Name	Description
	GetContentType (see page 3334)	Returns the value of the ContentType property.
	GetFieldName (see page 3334)	Returns the value of the FieldName property.
	GetFileName (see page 3335)	Returns the value of the FileName property.
	GetStream (see page 3335)	Returns the value of the Stream property.

Public Methods

	Name	Description
	Create (see page 3333)	Creates an initializes an instance of TWebRequestFile (see page 3332).
	Destroy (see page 3334)	Frees an instance of TWebRequestFile (see page 3332).

60.1.1.2 TWebRequestFile.Create Constructor

Creates an initializes an instance of TWebRequestFile (see page 3332).

Pascal

```
constructor Create(const AFieldName: AnsiString; const AFileName: AnsiString; const
AContentType: AnsiString; AContent: Pointer; AContentLength: Integer);
```

C++

```
__fastcall TWebRequestFile(const AnsiString AFieldName, const AnsiString AFileName, const
AnsiString AContentType, void * AContent, int AContentLength);
```

Description

Typically, applications do not call the TWebRequestFile (see page 3332) constructor. Instead, new Web request file objects are created by calling the Add method of a TWebRequestFiles (see page 3336) collection as the file uploads are extracted from the current Web request.

AFieldName is the name of the adapter field whose value this file represents. It becomes the value of the FieldName property.

AFileName is the name of the file this object represents. It becomes the value of the FileName property.

AContentType describes how the file content is encoded. It becomes the value of the ContentType property.

AContent is an array of bytes that is the content of the file. It is used to initialize the Stream property.

AContentLength is the number of bytes in AContent.

See Also

FieldName

FileName

ContentType

Stream

Add (see page 3338)

60.1.1.3 TWebRequestFile.Destroy Destructor

Frees an instance of TWebRequestFile (see page 3332).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TWebRequestFile();
```

Description

Do not free instances of TWebRequestFile (see page 3332). TWebRequestFile (see page 3332) objects are created and destroyed by the TWebRequestFiles (see page 3336) object that lists the uploaded files from the current request.

See Also

TWebRequestFiles (see page 3336)

60.1.1.4 TWebRequestFile Methods

The methods of the TWebRequestFile class are listed here.

Protected Methods

	Name	Description
	GetContentType (see page 3334)	Returns the value of the ContentType property.
	GetFieldName (see page 3334)	Returns the value of the FieldName property.
	GetFileName (see page 3335)	Returns the value of the FileName property.
	GetStream (see page 3335)	Returns the value of the Stream property.

60.1.1.4.1 TWebRequestFile.GetContentType Method

Returns the value of the ContentType property.

Pascal

```
function GetContentType: AnsiString; override;
```

C++

```
virtual __fastcall AnsiString GetContentType();
```

Description

GetContentType is the protected read implementation for the ContentType property.

In TAbstractWebRequestFile, GetContentType is abstract or, in C++ terminology, pure virtual, meaning it has no implementation. Descendant classes must override this method to provide an implementation for the ContentType property.

See Also

ContentType

60.1.1.4.2 TWebRequestFile.GetFieldName Method

Returns the value of the FieldName property.

Pascal

```
function GetFieldName: AnsiString; override;
```

C++

```
virtual __fastcall AnsiString GetFieldName();
```

Description

GetFieldName is the protected read implementation for the FieldName property.

In TAbstractWebRequestFile, GetFieldName is abstract or, in C++ terminology, pure virtual, meaning it has no implementation. Descendant classes must override this method to provide an implementation for the FieldName property.

See Also

FieldName

60.1.1.4.3 TWebRequestFile.GetFileName Method

Returns the value of the FileName property.

Pascal

```
function GetFileName: AnsiString; override;
```

C++

```
virtual __fastcall AnsiString GetFileName();
```

Description

GetFileName is the protected read implementation for the FileName property.

In TAbstractWebRequestFile, GetFileName is abstract or, in C++ terminology, pure virtual, meaning it has no implementation. Descendant classes must override this method to provide an implementation for the FileName property.

See Also

FileName

60.1.1.4.4 TWebRequestFile.GetStream Method

Returns the value of the Stream property.

Pascal

```
function GetStream: TStream; override;
```

C++

```
virtual __fastcall TStream GetStream();
```

Description

GetStream is the protected read implementation for the Stream property.

In TAbstractWebRequestFile, GetStream is abstract or, in C++ terminology, pure virtual, meaning it has no implementation. Descendant classes must override this method to provide an implementation for the Stream property.

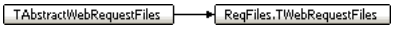
See Also

Stream

60.1.2 TWebRequestFiles Class

TWebRequestFiles stores a collection of TWebAbstractRequestFile objects.

Class Hierarchy



Pascal

```
TWebRequestFiles = class(TAbstractWebRequestFiles);
```

C++

```
class TWebRequestFiles : public TAbstractWebRequestFiles;
```

File

ReqFiles

Description

TWebRequestFiles manages a set of Web request files objects (TWebRequestFile (see page 3332) objects). Each Web request file object in the collection describes an uploaded file that was extracted from the current HTTP request object.

See Also

TWebRequestFile (see page 3332)

TWebRequest

60.1.2.1 TWebRequestFiles Members

The following tables list the members exposed by TWebRequestFiles.

Protected Methods

	Name	Description
	GetCount (see page 3338)	Returns the value of the Count property.
	GetItem (see page 3339)	Returns the value of the Items property.

Public Methods

	Name	Description
	Add (see page 3337)	This is the overview for the Add method overload.
	Create (see page 3336)	Creates and initializes an instance of TWebRequestFiles (see page 3336).
	Destroy (see page 3337)	Frees an instance of TWebRequestFiles (see page 3336).

60.1.2.2 TWebRequestFiles.Create Constructor

Creates and initializes an instance of TWebRequestFiles (see page 3336).

Pascal

```
constructor Create;
```

C++

```
__fastcall TWebRequestFiles();
```

Description

Call Create to instantiate TWebRequestFiles (see page 3336) at runtime. TWebRequestFiles (see page 3336) instances are created automatically by the Web request object to implement its Files property.

See Also

Files

60.1.2.3 TWebRequestFiles.Destroy Destructor

Frees an instance of TWebRequestFiles (see page 3336).

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TWebRequestFiles();
```

Description

Use the Free method to delete an instance of TWebRequestFiles (see page 3336). Free checks for a nil reference before calling Destroy.

See Also

System::TObject::Free

60.1.2.4 TWebRequestFiles Methods

The methods of the TWebRequestFiles class are listed here.

Protected Methods

	Name	Description
	GetCount (see page 3338)	Returns the value of the Count property.
	GetItem (see page 3339)	Returns the value of the Items property.

Public Methods

	Name	Description
	Add (see page 3337)	This is the overview for the Add method overload.

60.1.2.4.1 Add Method

This is the overview for the Add method overload.

Overload List

	Name	Description
	TWebRequestFiles.Add (AnsiString, AnsiString, AnsiString, Pointer, Integer) (see page 3338)	Adds a new Web request file object to the collection.
	TWebRequestFiles.Add (TWebRequestFile) (see page 3338)	Adds a new Web request file object to the collection.

60.1.2.4.1.1 TWebRequestFiles.Add Method (AnsiString, AnsiString, AnsiString, Pointer, Integer)

Adds a new Web request file object to the collection.

Pascal

```
procedure Add(const AName: AnsiString; const AFileName: AnsiString; const AContentType:
AnsiString; AContent: Pointer; AContentLength: Integer); overload;
procedure Add(AFile: TWebRequestFile); overload;
```

C++

```
__fastcall Add(const AnsiString AName, const AnsiString AFileName, const AnsiString
AContentType, void * AContent, int AContentLength);
__fastcall Add(TWebRequestFile AFile);
```

Description

Call Add to add a TWebRequestFile (see page 3332) object to the collection. Add inserts the new Web request file object at the end of the Items property array and increments the value of the Count property.

Use the first syntax to cause TWebRequestFiles (see page 3336) to create the Web request file object you are adding. Use the second syntax to add an existing TWebRequestFile (see page 3332) object. Using either syntax, TWebRequestFiles (see page 3336) assumes ownership of the new Web request file object and is responsible for freeing it.

AName is the name of the adapter field whose value the new file represents. It becomes the value of the FieldName property of the new TWebRequestFile (see page 3332) object.

AFileName is the name of the file to add. It becomes the value of the FileName property of the new TWebRequestFile (see page 3332) object.

AContentType describes how the file content is encoded. It becomes the value of the ContentType property of the new TWebRequestFile (see page 3332) object.

AContent is an array of bytes that is the content of the file. It is used to initialize the Stream property of the new TWebRequestFile (see page 3332) object.

AContentLength is the number of bytes in AContent.

AFile is an existing TWebRequestFile (see page 3332) object that describes the file you are adding.

See Also

Count

Items

FieldName

FileName

ContentType

Stream

60.1.2.4.2 TWebRequestFiles.GetCount Method

Returns the value of the Count property.

Pascal

```
function GetCount: Integer; override;
```

C++

```
virtual __fastcall int GetCount();
```

Description

GetCount is the protected read implementation of the Count property.

In TAbstractWebRequestFile, GetCount is abstract or, in C++ terminology, pure virtual, meaning it has no implementation. Descendant classes must override this method to provide an implementation for the Count property.

See Also

Count

60.1.2.4.3 TWebRequestFiles.GetItem Method

Returns the value of the Items property.

Pascal

```
function GetItem(I: Integer): TAbstractWebRequestFile; override;
```

C++

```
virtual __fastcall TAbstractWebRequestFile GetItem(int I);
```

Description

GetItem is the protected read implementation of the Items property.

I is the index of the Web request file object to return.

In TAbstractWebRequestFile, GetItem is abstract or, in C++ terminology, pure virtual, meaning it has no implementation. Descendant classes must override this method to provide an implementation for the Items property.

60.1.3 TWebRequestFileStream Class

Class Hierarchy

Pascal

```
TWebRequestFileStream = class(TCustomMemoryStream);
```

C++

```
class TWebRequestFileStream : public TCustomMemoryStream;
```

File

ReqFiles

Description

This is class ReqFiles.TWebRequestFileStream.

60.1.3.1 TWebRequestFileStream Members

The following tables list the members exposed by TWebRequestFileStream.

Public Methods

	Name	Description
	Create (see page 3340)	This is Create, a member of class TWebRequestFileStream.
V	Write (see page 3340)	Introduces a pure virtual method for writing to the stream.

60.1.3.2 TWebRequestFileStream.Create Constructor

Pascal

```
constructor Create(ABuffer: Pointer; Size: Integer);
```

C++

```
__fastcall TWebRequestFileStream(void * ABuffer, int Size);
```

Description

This is Create, a member of class TWebRequestFileStream.

60.1.3.3 TWebRequestFileStream Methods

The methods of the TWebRequestFileStream class are listed here.

Public Methods

	Name	Description
V	Write (see page 3340)	Introduces a pure virtual method for writing to the stream.

60.1.3.3.1 TWebRequestFileStream.Write Method

Introduces a pure virtual method for writing to the stream.

Pascal

```
function Write(const Buffer; Count: Longint): Longint; override;
```

C++

```
virtual __fastcall Longint Write(const  Buffer, Longint Count);
```

Description

Each descendant stream object defines a Write method that writes data to its particular storage medium (such as memory or a disk file). Write attempts to write up to Count bytes from Buffer, and returns the number of bytes actually written.

All the other data-writing methods of a stream (WriteBuffer, WriteComponent) call Write to do their actual writing.

See Also

WriteBuffer

WriteComponent

61 ReqMulti Namespace

This is namespace ReqMulti.

Topics

Name	Description
Classes (see page 3342)	The following table lists classes in this documentation.

Classes

	Name	Description
	TMultipartContentParser (see page 3342)	TMultipartContentParser extracts named value strings and uploaded files from an HTTP request sent by a multipart form.

61.1 Classes

The following table lists classes in this documentation.

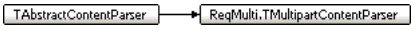
Classes

	Name	Description
	TMultipartContentParser (see page 3342)	TMultipartContentParser extracts named value strings and uploaded files from an HTTP request sent by a multipart form.

61.1.1 TMultipartContentParser Class

TMultipartContentParser extracts named value strings and uploaded files from an HTTP request sent by a multipart form.

Class Hierarchy



Pascal

```
TMultipartContentParser = class(TAbstractContentParser);
```

C++

```
class TMultipartContentParser : public TAbstractContentParser;
```

File

ReqMulti

Description

Web request objects use TMultipartContentParser object to parse the content of an HTTP request message from a multipart form when that request may contain uploaded files. The Web request uses this content parser to assign values for its ContentFields and Files properties.

TMultipartContentParser is only used when the current HTTP request object has a content type of 'multipart/form-data'. Multipart forms built using a WebSnap application automatically use this content type.

To allow your WebSnap application to use TMultipartContentParser in Delphi, add the ReqMulti (see page 3341) unit to the end of your project uses clause. In C++, include ReqMulti.hpp in the header of your project source file.

See Also

TWebRequest

TContentParser

61.1.1.1 TMultipartContentParser Members

The following tables list the members exposed by TMultipartContentParser.

Protected Methods

	Name	Description
	BufferContent (see page 3344)	This is BufferContent, a member of class TMultipartContentParser.

	ExtractContentTypeFields (see page 3344)	This is ExtractContentTypeFields, a member of class TMultipartContentParser.
	ExtractMultipartContentFields (see page 3345)	This is ExtractMultipartContentFields, a member of class TMultipartContentParser.
V	GetContentFields (see page 3345)	Returns the named value strings that are included in the content of the current Web request.
V	GetFiles (see page 3345)	Returns a set of uploaded files that the parser extracts from the current Web request.
	ParseMultiPart (see page 3346)	This is ParseMultiPart, a member of class TMultipartContentParser.
	ParseMultiPartContent (see page 3346)	This is ParseMultiPartContent, a member of class TMultipartContentParser.
	ParseMultipartHeaders (see page 3346)	This is ParseMultipartHeaders, a member of class TMultipartContentParser.

Public Methods

	Name	Description
V	CanParse (see page 3344)	Indicates whether a specified request message contains information encoded in a way that this parser understands.
V	Destroy (see page 3343)	Frees the memory associated with the TMultipartContentParser (see page 3342) object.

61.1.1.2 TMultipartContentParser.Destroy Destructor

Frees the memory associated with the TMultipartContentParser (see page 3342) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TMultipartContentParser();
```

Description

Do not free content parser instances. The Web request object frees the content parser when it no longer needs it.

61.1.1.3 TMultipartContentParser Methods

The methods of the TMultipartContentParser class are listed here.

Protected Methods

	Name	Description
	BufferContent (see page 3344)	This is BufferContent, a member of class TMultipartContentParser.
	ExtractContentTypeFields (see page 3344)	This is ExtractContentTypeFields, a member of class TMultipartContentParser.
	ExtractMultipartContentFields (see page 3345)	This is ExtractMultipartContentFields, a member of class TMultipartContentParser.
V	GetContentFields (see page 3345)	Returns the named value strings that are included in the content of the current Web request.
V	GetFiles (see page 3345)	Returns a set of uploaded files that the parser extracts from the current Web request.
	ParseMultiPart (see page 3346)	This is ParseMultiPart, a member of class TMultipartContentParser.
	ParseMultiPartContent (see page 3346)	This is ParseMultiPartContent, a member of class TMultipartContentParser.
	ParseMultipartHeaders (see page 3346)	This is ParseMultipartHeaders, a member of class TMultipartContentParser.

Public Methods

	Name	Description
V	CanParse (see page 3344)	Indicates whether a specified request message contains information encoded in a way that this parser understands.

61.1.1.3.1 TMultipartContentParser.BufferContent Method

Pascal

```
procedure BufferContent;
```

C++

```
__fastcall BufferContent();
```

Description

This is BufferContent, a member of class TMultipartContentParser.

61.1.1.3.2 TMultipartContentParser.CanParse Method

Indicates whether a specified request message contains information encoded in a way that this parser understands.

Pascal

```
class function CanParse(AWebRequest: TWebRequest): Boolean; override;
```

C++

```
virtual __fastcall Boolean CanParse(TWebRequest AWebRequest);
```

Description

CanParse indicates whether a content parser class can handle parsing a specified request message. CanParse is a class method (Delphi) or static method (C++), which means the Web request does not need an instantiated object to call this method. The Web request object calls the CanParse method of every registered content parser until it finds one that can parse its content.

AWebRequest is the current HTTP request message.

CanParse returns true if this class can parse AWebRequest, false if AWebRequest does not indicate an encoding that this class understands.

In TMultipartContentParser (see page 3342), CanParse returns true if AWebRequest has a content type of 'multipart/form-data'. If the Web request indicates some other content type, the Web request uses a different content parser class such as TContentParser.

See Also

ContentType

TContentParser

61.1.1.3.3 TMultipartContentParser.ExtractContentTypeFields Method

Pascal

```
procedure ExtractContentTypeFields(Strings: TStrings);
```

C++

```
__fastcall ExtractContentTypeFields(TStrings Strings);
```

Description

This is ExtractContentTypeFields, a member of class TMultipartContentParser.

61.1.1.3.4 TMultipartContentParser.ExtractMultipartContentFields Method

Pascal

```
procedure ExtractMultipartContentFields;
```

C++

```
__fastcall ExtractMultipartContentFields();
```

Description

This is ExtractMultipartContentFields, a member of class TMultipartContentParser.

61.1.1.3.5 TMultipartContentParser.GetContentFields Method

Returns the named value strings that are included in the content of the current Web request.

Pascal

```
function GetContentFields: TStrings; override;
```

C++

```
virtual __fastcall TStrings GetContentFields();
```

Description

GetContentFields returns a list of strings with the form "Name=Value" that represent any name/value pairs that the parser identifies in the content of the current Web request.

In TAbstractContentParser, GetContentFields is abstract or, in C++ terminology, pure virtual, meaning that this class does not supply an implementation. Descendant classes override this method to extract name/value pairs from the current request and add them to the returned string list.

Note: The content parser owns the string list it returns. The caller should not free the string list.

See Also

ContentFields

WebRequest

ExtractContentFields

61.1.1.3.6 TMultipartContentParser.GetFiles Method

Returns a set of uploaded files that the parser extracts from the current Web request.

Pascal

```
function GetFiles: TAbstractWebRequestFiles; override;
```

C++

```
virtual __fastcall TAbstractWebRequestFiles GetFiles();
```

Description

GetFiles returns a list of Web request file objects, each of which represents an uploaded file from the current Web request. Each Web request file object has properties to describe an uploaded file, including a stream object from which to read the content of the uploaded file.

In TAbstractContentParser, GetFiles is abstract or, in C++ terminology, pure virtual, meaning that this class does not supply an implementation. Descendant classes override this method to extract uploaded files from the current request and add them to the

Web request files object.

Note: The content parser owns the list it returns. The caller should not free this list.

See Also

Files

WebRequest

TAbstractWebRequestFile

61.1.1.3.7 TMultipartContentParser.ParseMultiPart Method

Pascal

```
procedure ParseMultiPart(Part: Pointer; Size: Integer);
```

C++

```
__fastcall ParseMultiPart(void * Part, int Size);
```

Description

This is ParseMultiPart, a member of class TMultipartContentParser.

61.1.1.3.8 TMultipartContentParser.ParseMultiPartContent Method

Pascal

```
procedure ParseMultiPartContent;
```

C++

```
__fastcall ParseMultiPartContent();
```

Description

This is ParseMultiPartContent, a member of class TMultipartContentParser.

61.1.1.3.9 TMultipartContentParser.ParseMultipartHeaders Method

Pascal

```
procedure ParseMultipartHeaders(Parser: THTTPParser; AContent: Pointer; AContentLength:
Integer);
```

C++

```
__fastcall ParseMultipartHeaders(THTTPParser Parser, void * AContent, int AContentLength);
```

Description

This is ParseMultipartHeaders, a member of class TMultipartContentParser.

62 Ribbon Namespace

This is namespace Ribbon.

Topics

Name	Description
Classes (see page 3349)	The following table lists classes in this documentation.
Interfaces (see page 3566)	The following table lists interfaces in this documentation.
Structs, Records, Enums (see page 3571)	The following table lists structs, records, enums in this documentation.
Types (see page 3576)	The following table lists types in this documentation.

Classes

	Name	Description
	ERibbonException (see page 3349)	This is class Ribbon.ERibbonException.
	TApplicationMenu (see page 3350)	Contains properties associated with the Application Menu (see page 3355).
	TCustomActionControlBar (see page 3355)	This is class Ribbon.TCustomActionControlBar.
	TCustomRibbon (see page 3358)	Base class for all ribbon controls.
	TCustomRibbonColorMap (see page 3401)	This is class Ribbon.TCustomRibbonColorMap.
	TCustomRibbonGroup (see page 3405)	Align and display all controls on ribbon.
	TCustomRibbonPage (see page 3425)	Holds groups of controls.
	TCustomRibbonQuickAccessPopupToolbar (see page 3438)	This is class Ribbon.TCustomRibbonQuickAccessPopupToolbar.
	TCustomRibbonQuickAccessToolbar (see page 3441)	Quck Access Toolbar base class.
	TCustomTabData (see page 3453)	Encapsulates all functions and status to manage scrolling of tabs and selected group.
	THelpButton (see page 3459)	Button to activate help.
	TOffice2007Button (see page 3461)	This is class Ribbon.TOffice2007Button.
	TQuickAccessToolbar (see page 3469)	Quick Access Toolbar appearance.
	TQuickAccessToolbarGroupItem (see page 3475)	Display Quick Access Toolbar commands to display entire Ribbon group.
	TQuickAccessToolbarGroups (see page 3477)	This is class Ribbon.TQuickAccessToolbarGroups.
	TRibbon (see page 3480)	Implement ribbon control.
	TRibbonCustomizeDlg (see page 3499)	This is class Ribbon.TRibbonCustomizeDlg.
	TRibbonGroup (see page 3500)	Control used to display the user selectable commands for the ribbon.
	TRibbonGroupCollapsedControl (see page 3506)	Control used to represent a collapsed group on a tab.
	TRibbonGroupDockForm (see page 3519)	Display a collapsed group.
	TRibbonGroupScrollButton (see page 3521)	This is class Ribbon.TRibbonGroupScrollButton.
	TRibbonLunaColorMap (see page 3525)	Color map class for Luna.
	TRibbonObsidianColorMap (see page 3530)	Color map class for Obsidian.

	TRibbonPage (see page 3536)	Tab page for ribbon control.
	TRibbonQuickAccessPopupToolbar (see page 3545)	This is class Ribbon.TRibbonQuickAccessPopupToolbar.
	TRibbonQuickAccessToolbar (see page 3546)	Quck Access Toolbar.
	TRibbonSilverColorMap (see page 3549)	Color map class for Silver.
	TRibbonTabItem (see page 3555)	This is class Ribbon.TRibbonTabItem.
	TRibbonTabs (see page 3560)	Collection of information about ribbon pages.

Interfaces

	Name	Description
	IRibbon (see page 3566)	Interface for ribbon control.
	IRibbonGroup (see page 3568)	Interface for any ribbon-specific group.
	IRibbonQuickAccessToolbar (see page 3569)	This is class Ribbon.IRibbonQuickAccessToolbar.

Structs, Records, Enums

	Name	Description
	TCommandType (see page 3571)	TCommandType enumerates the possible types of items to be displayed in the right pane of the Application Menu.
	TIconSize (see page 3572)	This is record Ribbon.TIconSize.
	TRibbonGroupVariant (see page 3572)	This is record Ribbon.TRibbonGroupVariant.
	TRibbonMetric (see page 3573)	This is record Ribbon.TRibbonMetric.
	TRibbonScrollButton (see page 3573)	This is record Ribbon.TRibbonScrollButton.
	TTabActiveScrollButton (see page 3574)	This is record Ribbon.TTabActiveScrollButton.
	TTabSeparatorState (see page 3574)	Position of Quick Access Toolbar.

Types

Name	Description
TCustomRibbonClass (see page 3576)	This is type Ribbon.TCustomRibbonClass.
TRecentItemClickEvent (see page 3576)	This is type Ribbon.TRecentItemClickEvent.
TRibbonScrollButtons (see page 3577)	This is type Ribbon.TRibbonScrollButtons.

62.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	ERibbonException (see page 3349)	This is class Ribbon.ERibbonException.
	TApplicationMenu (see page 3350)	Contains properties associated with the Application Menu (see page 3355).
	TCustomActionControlBar (see page 3355)	This is class Ribbon.TCustomActionControlBar.
	TCustomRibbon (see page 3358)	Base class for all ribbon controls.
	TCustomRibbonColorMap (see page 3401)	This is class Ribbon.TCustomRibbonColorMap.
	TCustomRibbonGroup (see page 3405)	Align and display all controls on ribbon.
	TCustomRibbonPage (see page 3425)	Holds groups of controls.
	TCustomRibbonQuickAccessPopupToolbar (see page 3438)	This is class Ribbon.TCustomRibbonQuickAccessPopupToolbar.
	TCustomRibbonQuickAccessToolbar (see page 3441)	Quck Access Toolbar base class.
	TCustomTabData (see page 3453)	Encapsulates all functions and status to manage scrolling of tabs and selected group.
	THelpButton (see page 3459)	Button to activate help.
	TOffice2007Button (see page 3461)	This is class Ribbon.TOffice2007Button.
	TQuickAccessToolbar (see page 3469)	Quick Access Toolbar appearance.
	TQuickAccessToolbarGroupItem (see page 3475)	Display Quick Access Toolbar commands to display entire Ribbon (see page 3347) group.
	TQuickAccessToolbarGroups (see page 3477)	This is class Ribbon.TQuickAccessToolbarGroups.
	TRibbon (see page 3480)	Implement ribbon control.
	TRibbonCustomizeDlg (see page 3499)	This is class Ribbon.TRibbonCustomizeDlg.
	TRibbonGroup (see page 3500)	Control used to display the user selectable commands for the ribbon.
	TRibbonGroupCollapsedControl (see page 3506)	Control used to represent a collapsed group on a tab.
	TRibbonGroupDockForm (see page 3519)	Display a collapsed group.
	TRibbonGroupScrollButton (see page 3521)	This is class Ribbon.TRibbonGroupScrollButton.
	TRibbonLunaColorMap (see page 3525)	Color map class for Luna.
	TRibbonObsidianColorMap (see page 3530)	Color map class for Obsidian.
	TRibbonPage (see page 3536)	Tab page for ribbon control.
	TRibbonQuickAccessPopupToolbar (see page 3545)	This is class Ribbon.TRibbonQuickAccessPopupToolbar.
	TRibbonQuickAccessToolbar (see page 3546)	Quck Access Toolbar.
	TRibbonSilverColorMap (see page 3549)	Color map class for Silver.
	TRibbonTabItem (see page 3555)	This is class Ribbon.TRibbonTabItem.
	TRibbonTabs (see page 3560)	Collection of information about ribbon pages.

62.1.1 ERibbonException Class

Class Hierarchy

Pascal

```
ERibbonException = class(Exception);
```

C++

```
class ERibbonException : public Exception;
```

File

Ribbon

Description

This is class Ribbon.ERibbonException.

62.1.2 TApplicationMenu Class

Contains properties associated with the Application Menu (see page 3355).

Class Hierarchy

Pascal

```
TApplicationMenu = class(TPersistent);
```

C++

```
class TApplicationMenu : public TPersistent;
```

File

Ribbon

Description

TApplicationMenu Contains properties associated with the Application Menu (see page 3355) on a ribbon.

Use the properties to change the appearance and behavior of the Application Menu (see page 3355) as displayed on the ribbon.

See Also

Caption (see page 3353)

Icon (see page 3354)

KeyTip (see page 3355)

62.1.2.1 TApplicationMenu Members

The following tables list the members exposed by TApplicationMenu.

Protected Methods

	Name	Description
	GetActionClientItem (see page 3352)	This is GetActionClientItem, a member of class TApplicationMenu.
	Initialize (see page 3353)	This is Initialize, a member of class TApplicationMenu.

Public Methods

	Name	Description
	Assign (see page 3352)	Copies the contents of another, similar object.
	Create (see page 3351)	This is Create, a member of class TApplicationMenu.
	Destroy (see page 3351)	Destroys the TPersistent instance and frees its memory.
	Show (see page 3353)	This is Show, a member of class TApplicationMenu.

Published Properties

	Name	Description
	Caption (see page 3353)	Text displayed above Application Menu (see page 3355).
	CommandType (see page 3354)	CommandType specifies the type of items displayed in the Application Menu (see page 3355).
	Icon (see page 3354)	Icon to display on the Application Button.
	IconSize (see page 3354)	This is IconSize, a member of class TApplicationMenu.
	KeyTip (see page 3355)	KeyTip displayed in KeyTip mode.
	Menu (see page 3355)	This is Menu, a member of class TApplicationMenu.
	ScreenTip (see page 3355)	This is ScreenTip, a member of class TApplicationMenu.

62.1.2.2 TApplicationMenu.Create Constructor

Pascal

```
constructor Create(ARibbon: TCustomRibbon); reintroduce;
```

C++

```
__fastcall TApplicationMenu(TCustomRibbon ARibbon);
```

Description

This is Create, a member of class TApplicationMenu.

62.1.2.3 TApplicationMenu.Destroy Destructor

Destroys the TPersistent instance and frees its memory.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TApplicationMenu();
```

Description

Do not call Destroy directly. Call Free instead. Free checks that the object reference is not nil before calling Destroy.

See Also

System::TObject::Free

System::TObject::Destroy

62.1.2.4 TApplicationMenu Methods

The methods of the TApplicationMenu class are listed here.

Protected Methods

	Name	Description
	GetActionClientItem (see page 3352)	This is GetActionClientItem, a member of class TApplicationMenu.
	Initialize (see page 3353)	This is Initialize, a member of class TApplicationMenu.

Public Methods

	Name	Description
	Assign (see page 3352)	Copies the contents of another, similar object.
	Show (see page 3353)	This is Show, a member of class TApplicationMenu.

62.1.2.4.1 TApplicationMenu.Assign Method

Copies the contents of another, similar object.

Pascal

```
procedure Assign(Source: TPersistent); override;
```

C++

```
virtual __fastcall Assign(TPersistent * Source);
```

Description

Call Assign to copy the properties or other attributes of one object from another. The standard form of a call to Assign is

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

which tells the Destination object to copy the contents of the Source object to itself.

Most objects override Assign to handle the assignment of properties from similar objects. When overriding Assign, call the inherited method if the destination object can't handle the assignment of properties from the class of the Source parameter.

If no overridden Assign method can handle the assignment of properties from Source, the method implemented in TPersistent calls the source object's AssignTo method. This allows the source object to handle the assignment. If the Source object is nil (Delphi) or NULL (C++), Assign raises an EConvertError exception.

In general, the statement

```
Destination := Source; {Delphi}
Destination = Source; // C++
```

is not the same as the statement

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

The assignment operator makes Destination reference the same object as Source, whereas the Assign method copies the contents of the object referenced by Source into the object referenced by Destination.

Note: The types of some properties are also objects. If these properties have write methods that use Assign to set the value of the property, then in these cases the assignment operator does the same thing as the Assign method.

See Also

AssignTo

EConvertError

62.1.2.4.2 TApplicationMenu.GetActionClientItem Method

Pascal

```
function GetActionClientItem: TActionClientItem;
```

C++

```
__fastcall TActionClientItem GetActionClientItem();
```

Description

This is GetActionClientItem, a member of class TApplicationMenu.

62.1.2.4.3 TApplicationMenu.Initialize Method

Pascal

```
procedure Initialize(AManager: TCustomScreenTipsManager);
```

C++

```
__fastcall Initialize(TCustomScreenTipsManager AManager);
```

Description

This is Initialize, a member of class TApplicationMenu.

62.1.2.4.4 TApplicationMenu.Show Method

Pascal

```
procedure Show(const Visible: Boolean);
```

C++

```
__fastcall Show(const Boolean Visible);
```

Description

This is Show, a member of class TApplicationMenu.

62.1.2.5 TApplicationMenu Properties

The properties of the TApplicationMenu class are listed here.

Published Properties

	Name	Description
	Caption (see page 3353)	Text displayed above Application Menu (see page 3355).
	CommandType (see page 3354)	CommandType specifies the type of items displayed in the Application Menu (see page 3355).
	Icon (see page 3354)	Icon to display on the Application Button.
	IconSize (see page 3354)	This is IconSize, a member of class TApplicationMenu.
	KeyTip (see page 3355)	KeyTip displayed in KeyTip mode.
	Menu (see page 3355)	This is Menu, a member of class TApplicationMenu.
	ScreenTip (see page 3355)	This is ScreenTip, a member of class TApplicationMenu.

62.1.2.5.1 TApplicationMenu.Caption Property

Text displayed above Application Menu (see page 3355).

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

Caption is the text displayed above the right panel in the Application Menu (see page 3355).

62.1.2.5.2 TApplicationMenu.CommandType Property

CommandType specifies the type of items displayed in the Application Menu (see page 3355).

Pascal

```
property CommandType: TCommandType;
```

C++

```
__property TCommandType CommandType;
```

Description

Use the CommandType property to set or check the type of items displayed in the right pane of the Application Menu (see page 3355). The following table lists the possible values for the CommandType property and the meaning of each value.

Value	Description
ctRecent	A list of recently used items will be displayed in the right pane of the application menu.
ctCommands	A list of commands will be displayed in the right pane of the application menu.

See Also

TApplicationMenu (see page 3350)

TCommandType (see page 3571)

62.1.2.5.3 TApplicationMenu.Icon Property

Icon to display on the Application Button.

Pascal

```
property Icon: TIcon;
```

C++

```
__property TIcon Icon;
```

Description

Icon is the icon to display on the Application Button. If no icon is assigned, the default application icon is used. The dimensions of the assigned icon are 24 x 24.

62.1.2.5.4 TApplicationMenu.IconSize Property

Pascal

```
property IconSize: TIconSize;
```

C++

```
__property TIconSize IconSize;
```

Description

This is IconSize, a member of class TApplicationMenu.

62.1.2.5.5 TApplicationMenu.KeyTip Property

KeyTip displayed in KeyTip mode.

Pascal

```
property KeyTip: string;
```

C++

```
__property UnicodeString KeyTip;
```

Description

KeyTip is the KeyTip that is displayed when the user has entered KeyTip mode.

62.1.2.5.6 TApplicationMenu.Menu Property

Pascal

```
property Menu: TRibbonApplicationMenuBar;
```

C++

```
__property TRibbonApplicationMenuBar Menu;
```

Description

This is Menu, a member of class TApplicationMenu.

62.1.2.5.7 TApplicationMenu.ScreenTip Property

Pascal

```
property ScreenTip: TScreenTipItem;
```

C++

```
__property TScreenTipItem ScreenTip;
```

Description

This is ScreenTip, a member of class TApplicationMenu.

62.1.3 TCustomActionControlBar Class

Class Hierarchy

Pascal

```
TCustomActionControlBar = class(TCustomActionToolbar);
```

C++

```
class TCustomActionControlBar : public TCustomActionToolbar;
```

File

Ribbon

Description

This is class Ribbon.TCustomActionControlBar.

62.1.3.1 TCustomActionControlBar Members

The following tables list the members exposed by TCustomActionControlBar.

Protected Methods

	Name	Description
	CMControlChange (see page 3357)	This is CMControlChange, a member of class TCustomActionControlBar.
V	CreateControls (see page 3357)	This is CreateControls, a member of class TCustomActionControlBar.
V	GetControlClass (see page 3357)	This is GetControlClass, a member of class TCustomActionControlBar.

Protected Properties

	Name	Description
R	CreatingControls (see page 3358)	This is CreatingControls, a member of class TCustomActionControlBar.

Public Methods

	Name	Description
V	Create (see page 3356)	This is Create, a member of class TCustomActionControlBar.
V	CreateControl (see page 3357)	This is CreateControl, a member of class TCustomActionControlBar.

62.1.3.2 TCustomActionControlBar.Create Constructor

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomActionControlBar(TComponent * AOwner);
```

Description

This is Create, a member of class TCustomActionControlBar.

62.1.3.3 TCustomActionControlBar Methods

The methods of the TCustomActionControlBar class are listed here.

Protected Methods

	Name	Description
	CMControlChange (see page 3357)	This is CMControlChange, a member of class TCustomActionControlBar.
V	CreateControls (see page 3357)	This is CreateControls, a member of class TCustomActionControlBar.
V	GetControlClass (see page 3357)	This is GetControlClass, a member of class TCustomActionControlBar.

Public Methods

	Name	Description
V	CreateControl (see page 3357)	This is CreateControl, a member of class TCustomActionControlBar.

62.1.3.3.1 TCustomActionControlBar.CMControlChange Method

Pascal

```
procedure CMControlChange(var Message: TCMControlChange); message CM_CONTROLCHANGE;
```

C++

```
__fastcall CMControlChange(TCMControlChange Message);
```

Description

This is CMControlChange, a member of class TCustomActionControlBar.

62.1.3.3.2 TCustomActionControlBar.CreateControl Method

Pascal

```
function CreateControl(AnItem: TActionClientItem): TCustomActionControl; override;
```

C++

```
virtual __fastcall TCustomActionControl CreateControl(TActionClientItem AnItem);
```

Description

This is CreateControl, a member of class TCustomActionControlBar.

62.1.3.3.3 TCustomActionControlBar.CreateControls Method

Pascal

```
procedure CreateControls; override;
```

C++

```
virtual __fastcall CreateControls();
```

Description

This is CreateControls, a member of class TCustomActionControlBar.

62.1.3.3.4 TCustomActionControlBar.GetControlClass Method

Pascal

```
function GetControlClass(AnItem: TActionClientItem): TCustomActionControlClass; override;
```

C++

```
virtual __fastcall TCustomActionControlClass GetControlClass(TActionClientItem AnItem);
```

Description

This is GetControlClass, a member of class TCustomActionControlBar.

62.1.3.4 TCustomActionControlBar Properties

The properties of the TCustomActionControlBar class are listed here.

Protected Properties

	Name	Description
	CreatingControls (see page 3358)	This is CreatingControls, a member of class TCustomActionControlBar.

62.1.3.4.1 TCustomActionControlBar.CreatingControls Property

Pascal

```
property CreatingControls: Boolean;
```

C++

```
__property Boolean CreatingControls;
```

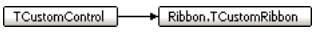
Description

This is CreatingControls, a member of class TCustomActionControlBar.

62.1.4 TCustomRibbon Class

Base class for all ribbon controls.

Class Hierarchy



Pascal

```
TCustomRibbon = class(TCustomControl);
```

C++

```
class TCustomRibbon : public TCustomControl;
```

File

Ribbon

Description

TCustomRibbon is the base class for all ribbon controls. A ribbon can hold controls, such as application menus, Quick Access Toolbars, and pages holding controls.

The following constants control ribbon appearance:

Value	Meaning
cRibbonHideWidth (see page 3365)	Parent width at which the Ribbon (see page 3347) must automatically hide.
cRibbonHideHeight (see page 3365)	Parent height at which the Ribbon (see page 3347) must automatically hide.
cRibbonQuickAccessToolbarTop (see page 3366)	Top position of the Quick Access Toolbar.
cRibbonQuickAccessToolbarLeft (see page 3366)	Left position of the Quick Access Toolbar.
cRibbonHeight (see page 3364)	Default height of the ribbon.
cRibbonQATHeight (see page 3366)	Height of the bottom aligned Quick Access Toolbar.
cRibbonUnthemedCaptionHeight (see page 3367)	Height of the caption section of the ribbon.

cRibbonFirstTabOffSet (see page 3364)	Position of the first tab that is displayed.
cRibbonTabSpacing (see page 3367)	Amount of spacing between tabs.
cRibbonTabHeight (see page 3366)	Height of the tabs.
cRibbonMinimizedHeight (see page 3365)	Minimized (see page 3397) height of the ribbon.

The following messages are used:

Value	Meaning
UM_DISPLAYKEYTIPS (see page 3368)	Message sent to display the KeyTips.
UM_CHECKSIZE (see page 3367)	Message sent after a `WMWindowPosChanged` (see page 3393) message.

See Also

Creating an Application that Uses Ribbon Controls

Adding Commands to the Ribbon

TApplicationMenu (see page 3350)

TCustomRibbonGroup (see page 3405)

TCustomRibbonPage (see page 3425)

TQuickAccessToolbar (see page 3469)

TRibbonGroup (see page 3500)

TRibbonPage (see page 3536)

62.1.4.1 TCustomRibbon Members

The following tables list the members exposed by TCustomRibbon.

Protected Events

	Name	Description
	OnLeftTabScrollButtonClick (see page 3400)	This is OnLeftTabScrollButtonClick, a member of class TCustomRibbon.
	OnRightTabScrollButtonClick (see page 3401)	This is OnRightTabScrollButtonClick, a member of class TCustomRibbon.

Protected Methods

	Name	Description
	CMBidimodechanged (see page 3371)	This is CMBidimodechanged, a member of class TCustomRibbon.
	CMDesignHitTest (see page 3371)	This is CMDesignHitTest, a member of class TCustomRibbon.
	CMFontchanged (see page 3371)	This is CMFontchanged, a member of class TCustomRibbon.
	CMMouseenter (see page 3371)	This is CMMouseenter, a member of class TCustomRibbon.
	CMMouseleave (see page 3372)	This is CMMouseleave, a member of class TCustomRibbon.
	CMTextchanged (see page 3372)	This is CMTextchanged, a member of class TCustomRibbon.
	CMThemeChanged (see page 3372)	This is CMThemeChanged, a member of class TCustomRibbon.
V	ContextMenuPopupHandler (see page 3372)	This is ContextMenuPopupHandler, a member of class TCustomRibbon.
V	CreateContextMenu (see page 3373)	This is CreateContextMenu, a member of class TCustomRibbon.
V	CustomFrameWndProc (see page 3373)	This is CustomFrameWndProc, a member of class TCustomRibbon.
V	DblClick (see page 3373)	OnDblClick event dispatcher.

	DefineProperties (see page 3374)	Provides methods to read and write the IsControl property to a stream such as a form file.
	DisplayKeyTips (see page 3374)	This is DisplayKeyTips, a member of class TCustomRibbon.
	DoAddToQuickAccessToolbar (see page 3375)	This is DoAddToQuickAccessToolbar, a member of class TCustomRibbon.
	DoChangeQuickAccessToolbarPosition (see page 3375)	This is DoChangeQuickAccessToolbarPosition, a member of class TCustomRibbon.
	DoContextPopup (see page 3375)	OnContextPopup event dispatcher.
	DoCustomizeQuickAccessToolbar (see page 3375)	This is DoCustomizeQuickAccessToolbar, a member of class TCustomRibbon.
	DoGroupStyleChanged (see page 3376)	This is DoGroupStyleChanged, a member of class TCustomRibbon.
	DoHelpButtonClick (see page 3376)	This is DoHelpButtonClick, a member of class TCustomRibbon.
	DoMinimizeRibbon (see page 3376)	This is DoMinimizeRibbon, a member of class TCustomRibbon.
	DoMouseWheelDown (see page 3376)	OnMouseWheelDown event dispatcher.
	DoMouseWheelUp (see page 3377)	OnMouseWheelUp event dispatcher.
	DoRecentItemClick (see page 3377)	This is DoRecentItemClick, a member of class TCustomRibbon.
	DoScrollTabsLeft (see page 3378)	This is DoScrollTabsLeft, a member of class TCustomRibbon.
	DoScrollTabsRight (see page 3378)	This is DoScrollTabsRight, a member of class TCustomRibbon.
	DoTabChange (see page 3378)	This is DoTabChange, a member of class TCustomRibbon.
	DrawCaption (see page 3378)	This is DrawCaption, a member of class TCustomRibbon.
	DrawQuickAccessToolbar (see page 3379)	This is DrawQuickAccessToolbar, a member of class TCustomRibbon.
	DrawSystemButtons (see page 3379)	This is DrawSystemButtons, a member of class TCustomRibbon.
	DrawTab (see page 3379)	This is DrawTab, a member of class TCustomRibbon.
	GetContextMenuOptions (see page 3379)	This is GetContextMenuOptions, a member of class TCustomRibbon.
	GetHelpButtonRect (see page 3380)	This is GetHelpButtonRect, a member of class TCustomRibbon.
	GetRibbonHeight (see page 3380)	This is GetRibbonHeight, a member of class TCustomRibbon.
	GetTabAt (see page 3381)	This is GetTabAt, a member of class TCustomRibbon.
	GetTabRect (see page 3381)	This is GetTabRect, a member of class TCustomRibbon.
	GetTabWidth (see page 3381)	This is GetTabWidth, a member of class TCustomRibbon.
	HasApplicationMenu (see page 3381)	This is HasApplicationMenu, a member of class TCustomRibbon.
	HasQuickAccessToolbar (see page 3382)	This is HasQuickAccessToolbar, a member of class TCustomRibbon.
	Idle (see page 3382)	This is Idle, a member of class TCustomRibbon.
	KeyDown (see page 3383)	Respond to key press events.
	Loaded (see page 3384)	Initializes the control after it is loaded from a stream.
	MouseDown (see page 3384)	OnMouseDown event dispatcher.
	MouseInSystemMenu (see page 3385)	This is MouseInSystemMenu, a member of class TCustomRibbon.
	MouseMove (see page 3385)	Respond to mouse moving over control area..
	MouseUp (see page 3385)	OnMouseUp event dispatcher.
	Notification (see page 3386)	Responds to notifications that components are being created or destroyed.
	Paint (see page 3387)	Renders the image of a custom control.
	RebuildRibbonMetrics (see page 3387)	This is RebuildRibbonMetrics, a member of class TCustomRibbon.
	Resize (see page 3388)	Respond to control resize.
	SetParent (see page 3389)	Sets the parent of the control.
	UMCheckResize (see page 3391)	This is UMCheckResize, a member of class TCustomRibbon.
	UMDisplayKeyTips (see page 3391)	This is UMDisplayKeyTips, a member of class TCustomRibbon.
	UpdateQATPosition (see page 3391)	This is UpdateQATPosition, a member of class TCustomRibbon.
	UpdateSystemMenuInfo (see page 3391)	This is UpdateSystemMenuInfo, a member of class TCustomRibbon.
	ValidTab (see page 3392)	This is ValidTab, a member of class TCustomRibbon.
	WMEraseBkgnd (see page 3392)	This is WMEraseBkgnd, a member of class TCustomRibbon.
	WMGetDlgCode (see page 3392)	This is WMGetDlgCode, a member of class TCustomRibbon.
	WMKillFocus (see page 3392)	This is WMKillFocus, a member of class TCustomRibbon.
	WMSettingChange (see page 3393)	This is WMSettingChange, a member of class TCustomRibbon.
	WMWindowPosChanged (see page 3393)	This is WMWindowPosChanged, a member of class TCustomRibbon.

Protected Properties

	Name	Description
	AllowKeyboardNavigation (see page 3395)	This is AllowKeyboardNavigation, a member of class TCustomRibbon.

	HideRibbon (see page 3396)	This is HideRibbon, a member of class TCustomRibbon.
	HotTabIndex (see page 3396)	This is HotTabIndex, a member of class TCustomRibbon.
	KeyTipsVisible (see page 3397)	This is KeyTipsVisible, a member of class TCustomRibbon.
R	Menu (see page 3397)	This is Menu, a member of class TCustomRibbon.
R	ScaleFactor (see page 3398)	This is ScaleFactor, a member of class TCustomRibbon.

Public Constants

	Name	Description
	cRibbonFirstTabOffSet (see page 3364)	This is cRibbonFirstTabOffSet, a member of class TCustomRibbon.
	cRibbonGroupCaptionHeight (see page 3364)	This is cRibbonGroupCaptionHeight, a member of class TCustomRibbon.
	cRibbonGroupHeight (see page 3364)	This is cRibbonGroupHeight, a member of class TCustomRibbon.
	cRibbonHeight (see page 3364)	This is cRibbonHeight, a member of class TCustomRibbon.
	cRibbonHideHeight (see page 3365)	This is cRibbonHideHeight, a member of class TCustomRibbon.
	cRibbonHideWidth (see page 3365)	This is cRibbonHideWidth, a member of class TCustomRibbon.
	cRibbonMinimizedHeight (see page 3365)	This is cRibbonMinimizedHeight, a member of class TCustomRibbon.
	cRibbonMinimumCaptionWidth (see page 3365)	This is cRibbonMinimumCaptionWidth, a member of class TCustomRibbon.
	cRibbonPageHeight (see page 3366)	This is cRibbonPageHeight, a member of class TCustomRibbon.
	cRibbonQATHeight (see page 3366)	This is cRibbonQATHeight, a member of class TCustomRibbon.
	cRibbonQuickAccessToolbarLeft (see page 3366)	This is cRibbonQuickAccessToolbarLeft, a member of class TCustomRibbon.
	cRibbonQuickAccessToolbarTop (see page 3366)	This is cRibbonQuickAccessToolbarTop, a member of class TCustomRibbon.
	cRibbonTabHeight (see page 3366)	This is cRibbonTabHeight, a member of class TCustomRibbon.
	cRibbonTabScrollButtonWidth (see page 3367)	This is cRibbonTabScrollButtonWidth, a member of class TCustomRibbon.
	cRibbonTabSpacing (see page 3367)	This is cRibbonTabSpacing, a member of class TCustomRibbon.
	cRibbonUnthemedCaptionHeight (see page 3367)	This is cRibbonUnthemedCaptionHeight, a member of class TCustomRibbon.
	UM_CHECKSIZE (see page 3367)	This is UM_CHECKSIZE, a member of class TCustomRibbon.
	UM_DISPLAYKEYTIPS (see page 3368)	This is UM_DISPLAYKEYTIPS, a member of class TCustomRibbon.

Public Events

	Name	Description
	OnHelpButtonClick (see page 3400)	This is OnHelpButtonClick, a member of class TCustomRibbon.
	OnRecentItemClick (see page 3401)	This is OnRecentItemClick, a member of class TCustomRibbon.
	OnTabChange (see page 3401)	This is OnTabChange, a member of class TCustomRibbon.

Public Methods

	Name	Description
	AddRecentItem (see page 3370)	Add new entry into Recent Items list.
	AddTab (see page 3370)	This is AddTab, a member of class TCustomRibbon.
	BeginUpdate (see page 3370)	This is BeginUpdate, a member of class TCustomRibbon.
	CheckRibbonSize (see page 3370)	This is CheckRibbonSize, a member of class TCustomRibbon.
	ClearRecentItems (see page 3371)	This is ClearRecentItems, a member of class TCustomRibbon.
V	Create (see page 3362)	Creates an instance of TCustomControl.
	CustomFrameSupported (see page 3373)	This is CustomFrameSupported, a member of class TCustomRibbon.
	DeleteTab (see page 3374)	Delete tab at specified Index.
V	Destroy (see page 3363)	Destroys an instance of TCustomControl.
	DoMouseWheelEx (see page 3377)	This is DoMouseWheelEx, a member of class TCustomRibbon.
	EndUpdate (see page 3379)	This is EndUpdate, a member of class TCustomRibbon.
V	GetIcon (see page 3380)	Return Icon used by Application Menu (see page 3397) Button.
	GetRibbonMetric (see page 3380)	This is GetRibbonMetric, a member of class TCustomRibbon.
	InvalidateHeaderArea (see page 3382)	Invalidate header section of Ribbon (see page 3347).
	InvalidateScrollButtons (see page 3382)	This is InvalidateScrollButtons, a member of class TCustomRibbon.
	InvalidateSystemMenuArea (see page 3383)	This is InvalidateSystemMenuArea, a member of class TCustomRibbon.
	InvalidateTabArea (see page 3383)	Invalidate area that displays tabs.
	MoveGroup (see page 3386)	Move group to new page.
	RemoveAllTabs (see page 3387)	Remove all Tabs (see page 3399) from Ribbon (see page 3347).
	RemoveRecentItem (see page 3388)	This is RemoveRecentItem, a member of class TCustomRibbon.

	RemoveTab (see page 3388)	Remove tab with specified caption.
	SetBounds (see page 3389)	Sets the windowed control's boundary properties all at once.
	TabHandledKeyTips (see page 3390)	Validate **KeyTip** input when **KeyTip**s State is **ktsTabs**.
	TabHideKeyTips (see page 3390)	Hide K**eyTip**s.
	TabShowKeyTips (see page 3390)	Display **KeyTip**s for **ktsTabs** State.

Public Nested Types

Name	Description
TTabChangeEvent (see page 3400)	This is nested type Ribbon.TCustomRibbon.TTabChangeEvent.

Public Properties

	Name	Description
	ActionManager (see page 3394)	This is ActionManager, a member of class TCustomRibbon.
	Active (see page 3394)	Indicates if form that ribbon is placed on is active form.
	ActivePage (see page 3394)	Page currently displayed by ribbon control.
	ApplicationMenu (see page 3395)	Change appearance of ribbon's associated TRibbonApplicationMenuBar (see page 3942) control.
	ColorMap (see page 3395)	This is ColorMap, a member of class TCustomRibbon.
	DocumentName (see page 3396)	Text in Ribbon (see page 3347) caption.
	HideTabs (see page 3396)	Shows or hides Tab portion of Ribbon (see page 3347).
	Minimized (see page 3397)	Show Ribbon (see page 3347) in minimized state where tabs can be selected but are not visible.
	QuickAccessToolbar (see page 3397)	Properties associated with TQuickAccessToolbar (see page 3469).
	ScreenTips (see page 3398)	This is ScreenTips, a member of class TCustomRibbon.
	ShowHelpButton (see page 3398)	Determines whether ribbon displays Help Button.
	Style (see page 3398)	This is Style, a member of class TCustomRibbon.
	TabIndex (see page 3399)	Change the active page of ribbon.
	Tabs (see page 3399)	Collection of TRibbonTabs (see page 3560) visible on ribbon.
	UseCustomFrame (see page 3399)	Indicates whether to use a custom frame for ribbon controls.

62.1.4.2 TCustomRibbon.Create Constructor

Creates an instance of TCustomControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomRibbon(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomControl. However, you should never attempt to instantiate a TCustomControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants. Controls placed on forms at design time are created automatically.

After calling the inherited constructor, Create initializes the control, including the following property settings:

Initializes a TCanvas object for its Canvas property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Create

Destroy

Canvas

Owner

62.1.4.3 TCustomRibbon.Destroy Destructor

Destroys an instance of TCustomControl.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomRibbon();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the control is not nil, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

As the control is destroyed, it:

Destroys the TCanvas object in its Canvas property.

Override Destroy to free any memory or resources allocated in the Create method. When declaring a Destroy method in a descendant control type, always add the override directive to the declaration and call the inherited Destroy as the last statement in the redeclared method.

When a control is created, Delphi automatically provides exception handling around the constructor call. The destructor is called if an exception escapes from the constructor. This means that the destructor code must be prepared to clean up a partially constructed instance. Check all data fields for zero before disposing of their contents.

See Also

System::TObject::Free

delete

Create

62.1.4.4 TCustomRibbon Constants

The constants of the TCustomRibbon class are listed here.

Public Constants

	Name	Description
	cRibbonFirstTabOffSet (see page 3364)	This is cRibbonFirstTabOffSet, a member of class TCustomRibbon.
	cRibbonGroupCaptionHeight (see page 3364)	This is cRibbonGroupCaptionHeight, a member of class TCustomRibbon.
	cRibbonGroupHeight (see page 3364)	This is cRibbonGroupHeight, a member of class TCustomRibbon.
	cRibbonHeight (see page 3364)	This is cRibbonHeight, a member of class TCustomRibbon.
	cRibbonHideHeight (see page 3365)	This is cRibbonHideHeight, a member of class TCustomRibbon.
	cRibbonHideWidth (see page 3365)	This is cRibbonHideWidth, a member of class TCustomRibbon.
	cRibbonMinimizedHeight (see page 3365)	This is cRibbonMinimizedHeight, a member of class TCustomRibbon.
	cRibbonMinimumCaptionWidth (see page 3365)	This is cRibbonMinimumCaptionWidth, a member of class TCustomRibbon.
	cRibbonPageHeight (see page 3366)	This is cRibbonPageHeight, a member of class TCustomRibbon.

	cRibbonQATHeight (see page 3366)	This is cRibbonQATHeight, a member of class TCustomRibbon.
	cRibbonQuickAccessToolbarLeft (see page 3366)	This is cRibbonQuickAccessToolbarLeft, a member of class TCustomRibbon.
	cRibbonQuickAccessToolbarTop (see page 3366)	This is cRibbonQuickAccessToolbarTop, a member of class TCustomRibbon.
	cRibbonTabHeight (see page 3366)	This is cRibbonTabHeight, a member of class TCustomRibbon.
	cRibbonTabScrollButtonWidth (see page 3367)	This is cRibbonTabScrollButtonWidth, a member of class TCustomRibbon.
	cRibbonTabSpacing (see page 3367)	This is cRibbonTabSpacing, a member of class TCustomRibbon.
	cRibbonUnthemedCaptionHeight (see page 3367)	This is cRibbonUnthemedCaptionHeight, a member of class TCustomRibbon.
	UM_CHECKSIZE (see page 3367)	This is UM_CHECKSIZE, a member of class TCustomRibbon.
	UM_DISPLAYKEYTIPS (see page 3368)	This is UM_DISPLAYKEYTIPS, a member of class TCustomRibbon.

62.1.4.4.1 TCustomRibbon.cRibbonFirstTabOffSet Constant

Pascal

```
const cRibbonFirstTabOffSet = 47;
```

C++

```
cRibbonFirstTabOffSet = 47;
```

Description

This is cRibbonFirstTabOffSet, a member of class TCustomRibbon.

62.1.4.4.2 TCustomRibbon.cRibbonGroupCaptionHeight Constant

Pascal

```
const cRibbonGroupCaptionHeight = 16;
```

C++

```
cRibbonGroupCaptionHeight = 16;
```

Description

This is cRibbonGroupCaptionHeight, a member of class TCustomRibbon.

62.1.4.4.3 TCustomRibbon.cRibbonGroupHeight Constant

Pascal

```
const cRibbonGroupHeight = 86;
```

C++

```
cRibbonGroupHeight = 86;
```

Description

This is cRibbonGroupHeight, a member of class TCustomRibbon.

62.1.4.4.4 TCustomRibbon.cRibbonHeight Constant

Pascal

```
const cRibbonHeight = 117;
```

C++

```
cRibbonHeight = 117;
```

Description

This is cRibbonHeight, a member of class TCustomRibbon.

62.1.4.4.5 TCustomRibbon.cRibbonHideHeight Constant

Pascal

```
const cRibbonHideHeight = 250;
```

C++

```
cRibbonHideHeight = 250;
```

Description

This is cRibbonHideHeight, a member of class TCustomRibbon.

62.1.4.4.6 TCustomRibbon.cRibbonHideWidth Constant

Pascal

```
const cRibbonHideWidth = 300;
```

C++

```
cRibbonHideWidth = 300;
```

Description

This is cRibbonHideWidth, a member of class TCustomRibbon.

62.1.4.4.7 TCustomRibbon.cRibbonMinimizedHeight Constant

Pascal

```
const cRibbonMinimizedHeight = 27;
```

C++

```
cRibbonMinimizedHeight = 27;
```

Description

This is cRibbonMinimizedHeight, a member of class TCustomRibbon.

62.1.4.4.8 TCustomRibbon.cRibbonMinimumCaptionWidth Constant

Pascal

```
const cRibbonMinimumCaptionWidth = 50;
```

C++

```
cRibbonMinimumCaptionWidth = 50;
```

Description

This is cRibbonMinimumCaptionWidth, a member of class TCustomRibbon.

62.1.4.4.9 TCustomRibbon.cRibbonPageHeight Constant

Pascal

```
const cRibbonPageHeight = 93;
```

C++

```
cRibbonPageHeight = 93;
```

Description

This is cRibbonPageHeight, a member of class TCustomRibbon.

62.1.4.4.10 TCustomRibbon.cRibbonQATHeight Constant

Pascal

```
const cRibbonQATHeight = 26;
```

C++

```
cRibbonQATHeight = 26;
```

Description

This is cRibbonQATHeight, a member of class TCustomRibbon.

62.1.4.4.11 TCustomRibbon.cRibbonQuickAccessToolbarLeft Constant

Pascal

```
const cRibbonQuickAccessToolbarLeft = 34;
```

C++

```
cRibbonQuickAccessToolbarLeft = 34;
```

Description

This is cRibbonQuickAccessToolbarLeft, a member of class TCustomRibbon.

62.1.4.4.12 TCustomRibbon.cRibbonQuickAccessToolbarTop Constant

Pascal

```
const cRibbonQuickAccessToolbarTop = 5;
```

C++

```
cRibbonQuickAccessToolbarTop = 5;
```

Description

This is cRibbonQuickAccessToolbarTop, a member of class TCustomRibbon.

62.1.4.4.13 TCustomRibbon.cRibbonTabHeight Constant

Pascal

```
const cRibbonTabHeight = 23;
```

C++

```
cRibbonTabHeight = 23;
```

Description

This is cRibbonTabHeight, a member of class TCustomRibbon.

62.1.4.4.14 TCustomRibbon.cRibbonTabScrollButtonWidth Constant

Pascal

```
const cRibbonTabScrollButtonWidth = 12;
```

C++

```
cRibbonTabScrollButtonWidth = 12;
```

Description

This is cRibbonTabScrollButtonWidth, a member of class TCustomRibbon.

62.1.4.4.15 TCustomRibbon.cRibbonTabSpacing Constant

Pascal

```
const cRibbonTabSpacing = 6;
```

C++

```
cRibbonTabSpacing = 6;
```

Description

This is cRibbonTabSpacing, a member of class TCustomRibbon.

62.1.4.4.16 TCustomRibbon.cRibbonUnthemedCaptionHeight Constant

Pascal

```
const cRibbonUnthemedCaptionHeight = 30;
```

C++

```
cRibbonUnthemedCaptionHeight = 30;
```

Description

This is cRibbonUnthemedCaptionHeight, a member of class TCustomRibbon.

62.1.4.4.17 TCustomRibbon.UM_CHECKSIZE Constant

Pascal

```
const UM_CHECKSIZE = WM_USER + 2;
```

C++

```
UM_CHECKSIZE = WM_USER + 2;
```

Description

This is UM_CHECKSIZE, a member of class TCustomRibbon.

62.1.4.4.18 TCustomRibbon.UM_DISPLAYKEYTIPS Constant

Pascal

```
const UM_DISPLAYKEYTIPS = WM_USER + 1;
```

C++

```
UM_DISPLAYKEYTIPS = WM_USER + 1;
```

Description

This is UM_DISPLAYKEYTIPS, a member of class TCustomRibbon.

62.1.4.5 TCustomRibbon Methods

The methods of the TCustomRibbon class are listed here.

Protected Methods

	Name	Description
	CMBidimodechanged (see page 3371)	This is CMBidimodechanged, a member of class TCustomRibbon.
	CMDesignHitTest (see page 3371)	This is CMDesignHitTest, a member of class TCustomRibbon.
	CMFontchanged (see page 3371)	This is CMFontchanged, a member of class TCustomRibbon.
	CMMouseenter (see page 3371)	This is CMMouseenter, a member of class TCustomRibbon.
	CMMouseleave (see page 3372)	This is CMMouseleave, a member of class TCustomRibbon.
	CMTextchanged (see page 3372)	This is CMTextchanged, a member of class TCustomRibbon.
	CMThemeChanged (see page 3372)	This is CMThemeChanged, a member of class TCustomRibbon.
	ContextMenuPopupHandler (see page 3372)	This is ContextMenuPopupHandler, a member of class TCustomRibbon.
	CreateContextMenu (see page 3373)	This is CreateContextMenu, a member of class TCustomRibbon.
	CustomFrameWndProc (see page 3373)	This is CustomFrameWndProc, a member of class TCustomRibbon.
	DblClick (see page 3373)	OnDblClick event dispatcher.
	DefineProperties (see page 3374)	Provides methods to read and write the IsControl property to a stream such as a form file.
	DisplayKeyTips (see page 3374)	This is DisplayKeyTips, a member of class TCustomRibbon.
	DoAddToQuickAccessToolbar (see page 3375)	This is DoAddToQuickAccessToolbar, a member of class TCustomRibbon.
	DoChangeQuickAccessToolbarPosition (see page 3375)	This is DoChangeQuickAccessToolbarPosition, a member of class TCustomRibbon.
	DoContextPopup (see page 3375)	OnContextPopup event dispatcher.
	DoCustomizeQuickAccessToolbar (see page 3375)	This is DoCustomizeQuickAccessToolbar, a member of class TCustomRibbon.
	DoGroupStyleChanged (see page 3376)	This is DoGroupStyleChanged, a member of class TCustomRibbon.
	DoHelpButtonClick (see page 3376)	This is DoHelpButtonClick, a member of class TCustomRibbon.
	DoMinimizeRibbon (see page 3376)	This is DoMinimizeRibbon, a member of class TCustomRibbon.
	DoMouseWheelDown (see page 3376)	OnMouseWheelDown event dispatcher.
	DoMouseWheelUp (see page 3377)	OnMouseWheelUp event dispatcher.
	DoRecentItemClick (see page 3377)	This is DoRecentItemClick, a member of class TCustomRibbon.
	DoScrollTabsLeft (see page 3378)	This is DoScrollTabsLeft, a member of class TCustomRibbon.
	DoScrollTabsRight (see page 3378)	This is DoScrollTabsRight, a member of class TCustomRibbon.
	DoTabChange (see page 3378)	This is DoTabChange, a member of class TCustomRibbon.
	DrawCaption (see page 3378)	This is DrawCaption, a member of class TCustomRibbon.
	DrawQuickAccessToolbar (see page 3379)	This is DrawQuickAccessToolbar, a member of class TCustomRibbon.
	DrawSystemButtons (see page 3379)	This is DrawSystemButtons, a member of class TCustomRibbon.
	DrawTab (see page 3379)	This is DrawTab, a member of class TCustomRibbon.
	GetContextMenuOptions (see page 3379)	This is GetContextMenuOptions, a member of class TCustomRibbon.
	GetHelpButtonRect (see page 3380)	This is GetHelpButtonRect, a member of class TCustomRibbon.
	GetRibbonHeight (see page 3380)	This is GetRibbonHeight, a member of class TCustomRibbon.
	GetTabAt (see page 3381)	This is GetTabAt, a member of class TCustomRibbon.

	GetTabRect (see page 3381)	This is GetTabRect, a member of class TCustomRibbon.
	GetTabWidth (see page 3381)	This is GetTabWidth, a member of class TCustomRibbon.
	HasApplicationMenu (see page 3381)	This is HasApplicationMenu, a member of class TCustomRibbon.
	HasQuickAccessToolbar (see page 3382)	This is HasQuickAccessToolbar, a member of class TCustomRibbon.
	Idle (see page 3382)	This is Idle, a member of class TCustomRibbon.
	KeyDown (see page 3383)	Respond to key press events.
	Loaded (see page 3384)	Initializes the control after it is loaded from a stream.
	MouseDown (see page 3384)	OnMouseDown event dispatcher.
	MouseInSystemMenu (see page 3385)	This is MouseInSystemMenu, a member of class TCustomRibbon.
	MouseMove (see page 3385)	Respond to mouse moving over control area..
	MouseUp (see page 3385)	OnMouseUp event dispatcher.
	Notification (see page 3386)	Responds to notifications that components are being created or destroyed.
	Paint (see page 3387)	Renders the image of a custom control.
	RebuildRibbonMetrics (see page 3387)	This is RebuildRibbonMetrics, a member of class TCustomRibbon.
	Resize (see page 3388)	Respond to control resize.
	SetParent (see page 3389)	Sets the parent of the control.
	UMCheckResize (see page 3391)	This is UMCheckResize, a member of class TCustomRibbon.
	UMDisplayKeyTips (see page 3391)	This is UMDisplayKeyTips, a member of class TCustomRibbon.
	UpdateQATPosition (see page 3391)	This is UpdateQATPosition, a member of class TCustomRibbon.
	UpdateSystemMenuInfo (see page 3391)	This is UpdateSystemMenuInfo, a member of class TCustomRibbon.
	ValidTab (see page 3392)	This is ValidTab, a member of class TCustomRibbon.
	WMEraseBkgnd (see page 3392)	This is WMEraseBkgnd, a member of class TCustomRibbon.
	WMGetDlgCode (see page 3392)	This is WMGetDlgCode, a member of class TCustomRibbon.
	WMKillFocus (see page 3392)	This is WMKillFocus, a member of class TCustomRibbon.
	WMSettingChange (see page 3393)	This is WMSettingChange, a member of class TCustomRibbon.
	WMWindowPosChanged (see page 3393)	This is WMWindowPosChanged, a member of class TCustomRibbon.

Public Methods

	Name	Description
	AddRecentItem (see page 3370)	Add new entry into Recent Items list.
	AddTab (see page 3370)	This is AddTab, a member of class TCustomRibbon.
	BeginUpdate (see page 3370)	This is BeginUpdate, a member of class TCustomRibbon.
	CheckRibbonSize (see page 3370)	This is CheckRibbonSize, a member of class TCustomRibbon.
	ClearRecentItems (see page 3371)	This is ClearRecentItems, a member of class TCustomRibbon.
	CustomFrameSupported (see page 3373)	This is CustomFrameSupported, a member of class TCustomRibbon.
	DeleteTab (see page 3374)	Delete tab at specified Index.
	DoMouseWheelEx (see page 3377)	This is DoMouseWheelEx, a member of class TCustomRibbon.
	EndUpdate (see page 3379)	This is EndUpdate, a member of class TCustomRibbon.
	GetIcon (see page 3380)	Return Icon used by Application Menu (see page 3397) Button.
	GetRibbonMetric (see page 3380)	This is GetRibbonMetric, a member of class TCustomRibbon.
	InvalidateHeaderArea (see page 3382)	Invalidate header section of Ribbon (see page 3347).
	InvalidateScrollButtons (see page 3382)	This is InvalidateScrollButtons, a member of class TCustomRibbon.
	InvalidateSystemMenuArea (see page 3383)	This is InvalidateSystemMenuArea, a member of class TCustomRibbon.
	InvalidateTabArea (see page 3383)	Invalidate area that displays tabs.
	MoveGroup (see page 3386)	Move group to new page.
	RemoveAllTabs (see page 3387)	Remove all Tabs (see page 3399) from Ribbon (see page 3347).
	RemoveRecentItem (see page 3388)	This is RemoveRecentItem, a member of class TCustomRibbon.
	RemoveTab (see page 3388)	Remove tab with specified caption.
	SetBounds (see page 3389)	Sets the windowed control's boundary properties all at once.
	TabHandledKeyTips (see page 3390)	Validate **KeyTip** input when **KeyTip**s State is **ktsTabs**.
	TabHideKeyTips (see page 3390)	Hide K**eyTip**s.
	TabShowKeyTips (see page 3390)	Display **KeyTip**s for **ktsTabs** State.

62.1.4.5.1 TCustomRibbon.AddRecentItem Method

Add new entry into Recent Items list.

Pascal

```
function AddRecentItem(const FileName: string): Integer;
```

C++

```
__fastcall int AddRecentItem(const UnicodeString FileName);
```

Description

Call AddRecentItem to add a new entry into the Recent Items list to the right of the Application Menu (see page 3397). The new entry is automatically added to the top of the Recent items list, and if an entry with the same filename already exists, it is deleted.

This method should only be called when the command type for the application menu is set to **ctRecent**.

62.1.4.5.2 TCustomRibbon.AddTab Method

Pascal

```
function AddTab(const APage: TCustomRibbonPage): TRibbonTabItem;
```

C++

```
__fastcall TRibbonTabItem AddTab(const TCustomRibbonPage APage);
```

Description

This is AddTab, a member of class TCustomRibbon.

62.1.4.5.3 TCustomRibbon.BeginUpdate Method

Pascal

```
procedure BeginUpdate;
```

C++

```
__fastcall BeginUpdate();
```

Description

This is BeginUpdate, a member of class TCustomRibbon.

62.1.4.5.4 TCustomRibbon.CheckRibbonSize Method

Pascal

```
procedure CheckRibbonSize;
```

C++

```
__fastcall CheckRibbonSize();
```

Description

This is CheckRibbonSize, a member of class TCustomRibbon.

62.1.4.5.5 TCustomRibbon.ClearRecentItems Method

Pascal

```
procedure ClearRecentItems;
```

C++

```
__fastcall ClearRecentItems();
```

Description

This is ClearRecentItems, a member of class TCustomRibbon.

62.1.4.5.6 TCustomRibbon.CMBidimodechanged Method

Pascal

```
procedure CMBidimodechanged(var Message: TMessage); message CM_BIDIMODECHANGED;
```

C++

```
__fastcall CMBidimodechanged(TMessage Message);
```

Description

This is CMBidimodechanged, a member of class TCustomRibbon.

62.1.4.5.7 TCustomRibbon.CMDesignHitTest Method

Pascal

```
procedure CMDesignHitTest(var Message: TCMDesignHitTest); message CM_DESIGNHITTEST;
```

C++

```
__fastcall CMDesignHitTest(TCMDesignHitTest Message);
```

Description

This is CMDesignHitTest, a member of class TCustomRibbon.

62.1.4.5.8 TCustomRibbon.CMFontchanged Method

Pascal

```
procedure CMFontchanged(var Message: TMessage); message CM_FONTCHANGED;
```

C++

```
__fastcall CMFontchanged(TMessage Message);
```

Description

This is CMFontchanged, a member of class TCustomRibbon.

62.1.4.5.9 TCustomRibbon.CMMouseenter Method

Pascal

```
procedure CMMouseenter(var Message: TMessage); message CM_MOUSEENTER;
```

C++

```
__fastcall CMMouseenter(TMessage Message);
```

Description

This is CMMouseenter, a member of class TCustomRibbon.

62.1.4.5.10 TCustomRibbon.CMMouseleave Method

Pascal

```
procedure CMMouseleave(var Message: TMessage); message CM_MOUSELEAVE;
```

C++

```
__fastcall CMMouseleave(TMessage Message);
```

Description

This is CMMouseleave, a member of class TCustomRibbon.

62.1.4.5.11 TCustomRibbon.CMTextchanged Method

Pascal

```
procedure CMTextchanged(var Message: TMessage); message CM_TEXTCHANGED;
```

C++

```
__fastcall CMTextchanged(TMessage Message);
```

Description

This is CMTextchanged, a member of class TCustomRibbon.

62.1.4.5.12 TCustomRibbon.CMThemeChanged Method

Pascal

```
procedure CMThemeChanged(var Message: TMessage); message CM_THEMECHANGED;
```

C++

```
__fastcall CMThemeChanged(TMessage Message);
```

Description

This is CMThemeChanged, a member of class TCustomRibbon.

62.1.4.5.13 TCustomRibbon.ContextMenuPopupHandler Method

Pascal

```
procedure ContextMenuPopupHandler(Sender: TObject); virtual;
```

C++

```
virtual __fastcall ContextMenuPopupHandler(TObject * Sender);
```

Description

This is ContextMenuPopupHandler, a member of class TCustomRibbon.

62.1.4.5.14 TCustomRibbon.CreateContextMenu Method

Pascal

```
procedure CreateContextMenu; virtual;
```

C++

```
virtual __fastcall CreateContextMenu();
```

Description

This is CreateContextMenu, a member of class TCustomRibbon.

62.1.4.5.15 TCustomRibbon.CustomFrameSupported Method

Pascal

```
function CustomFrameSupported: Boolean;
```

C++

```
__fastcall Boolean CustomFrameSupported();
```

Description

This is CustomFrameSupported, a member of class TCustomRibbon.

62.1.4.5.16 TCustomRibbon.CustomFrameWndProc Method

Pascal

```
procedure CustomFrameWndProc(var Message: TMessage); virtual;
```

C++

```
virtual __fastcall CustomFrameWndProc(TMessage Message);
```

Description

This is CustomFrameWndProc, a member of class TCustomRibbon.

62.1.4.5.17 TCustomRibbon.DblClick Method

OnDblClick event dispatcher.

Pascal

```
procedure DblClick; override;
```

C++

```
virtual __fastcall DblClick();
```

Description

DblClick calls any event handler attached to the OnDblClick event. Override DblClick to provide other responses in addition to the inherited event-handler call when the user double-clicks the control.

The DblClick method is triggered by left mouse-button double-click message (WM_LBUTTONDBLCLK) from Windows.

See Also

Click

OnDblClick

62.1.4.5.18 TCustomRibbon.DefineProperties Method

Provides methods to read and write the IsControl property to a stream such as a form file.

Pascal

```
procedure DefineProperties(Filer: TFiler); override;
```

C++

```
virtual __fastcall DefineProperties(TFiler Filer);
```

Description

DefineProperties is called automatically by the streaming system that loads and saves VCL components. It allows persistent objects to read and write properties that are not published. Published properties are loaded and stored automatically.

DefineProperties uses the methods of the filer passed as the Filer parameter to assign methods that can load and save the IsControl property.

Override DefineProperties to persist unpublished properties. Call the DefineBinaryProperty or DefineProperty method of the Filer parameter to assign methods that load or save property values.

Note: When overriding this method, call the parent class's DefineProperties method first.

See Also

DefineBinaryProperty

DefineProperty

DefineProperties

IsControl

62.1.4.5.19 TCustomRibbon.DeleteTab Method

Delete tab at specified Index.

Pascal

```
procedure DeleteTab(const Index: Integer);
```

C++

```
__fastcall DeleteTab(const int Index);
```

Description

DeleteTab deletes the tab at the specified Index. The Page and any associated Groups are also deleted

62.1.4.5.20 TCustomRibbon.DisplayKeyTips Method

Pascal

```
procedure DisplayKeyTips(const State: TKeyTipState); virtual;
```

C++

```
virtual __fastcall DisplayKeyTips(const TKeyTipState State);
```

Description

This is DisplayKeyTips, a member of class TCustomRibbon.

62.1.4.5.21 TCustomRibbon.DoAddToQuickAccessToolbar Method

Pascal

```
procedure DoAddToQuickAccessToolbar; virtual;
```

C++

```
virtual __fastcall DoAddToQuickAccessToolbar();
```

Description

This is DoAddToQuickAccessToolbar, a member of class TCustomRibbon.

62.1.4.5.22 TCustomRibbon.DoChangeQuickAccessToolbarPosition Method

Pascal

```
procedure DoChangeQuickAccessToolbarPosition; virtual;
```

C++

```
virtual __fastcall DoChangeQuickAccessToolbarPosition();
```

Description

This is DoChangeQuickAccessToolbarPosition, a member of class TCustomRibbon.

62.1.4.5.23 TCustomRibbon.DoContextPopup Method

OnContextPopup event dispatcher.

Pascal

```
procedure DoContextPopup(MousePos: TPoint; var Handled: Boolean); override;
```

C++

```
virtual __fastcall DoContextPopup(TPoint MousePos, Boolean Handled);
```

Description

OnContextPopup calls any event handler attached to the OnContextPopup event. When creating a component that descends from TControl, override OnContextPopup to provide other responses in addition to the inherited event-handler call when the user invokes the control's context menu.

MousePos indicates the position where the user right-clicked the control. If the user invoked the context menu using the keyboard, MousePos is (-1,-1).

Handled returns a Boolean value indicating whether the control should continue with its default response to the user action.

See Also

OnContextPopup

62.1.4.5.24 TCustomRibbon.DoCustomizeQuickAccessToolbar Method

Pascal

```
procedure DoCustomizeQuickAccessToolbar; virtual;
```

C++

```
virtual __fastcall DoCustomizeQuickAccessToolbar();
```

Description

This is DoCustomizeQuickAccessToolbar, a member of class TCustomRibbon.

62.1.4.5.25 TCustomRibbon.DoGroupStyleChanged Method

Pascal

```
procedure DoGroupStyleChanged; virtual;
```

C++

```
virtual __fastcall DoGroupStyleChanged();
```

Description

This is DoGroupStyleChanged, a member of class TCustomRibbon.

62.1.4.5.26 TCustomRibbon.DoHelpButtonClick Method

Pascal

```
procedure DoHelpButtonClick; virtual;
```

C++

```
virtual __fastcall DoHelpButtonClick();
```

Description

This is DoHelpButtonClick, a member of class TCustomRibbon.

62.1.4.5.27 TCustomRibbon.DoMinimizeRibbon Method

Pascal

```
procedure DoMinimizeRibbon; virtual;
```

C++

```
virtual __fastcall DoMinimizeRibbon();
```

Description

This is DoMinimizeRibbon, a member of class TCustomRibbon.

62.1.4.5.28 TCustomRibbon.DoMouseWheelDown Method

OnMouseWheelDown event dispatcher.

Pascal

```
function DoMouseWheelDown(Shift: TShiftState; MousePos: TPoint): Boolean; override;
```

C++

```
virtual __fastcall Boolean DoMouseWheelDown(TShiftState Shift, TPoint MousePos);
```

Description

The DoMouseWheel method calls DoMouseWheelDown to generate an OnMouseWheelDown event.

The Shift parameter indicates the state of the Shift, Alt, and Ctrl keys.

The MousePos parameter indicates the position of the mouse pointer.

DoMouseWheelDown returns true if an event handler indicates that it handled the message, and false otherwise.

Override DoMouseWheelDown to perform class-specific tasks in addition to calling the event handler when the mouse wheel is rotated downward.

See Also

OnMouseWheelDown

DoMouseWheel

62.1.4.5.29 TCustomRibbon.DoMouseWheelEx Method

Pascal

```
function DoMouseWheelEx(Shift: TShiftState; WheelDelta: Integer; MousePos: TPoint): Boolean;
```

C++

```
__fastcall Boolean DoMouseWheelEx(TShiftState Shift, int WheelDelta, TPoint MousePos);
```

Description

This is DoMouseWheelEx, a member of class TCustomRibbon.

62.1.4.5.30 TCustomRibbon.DoMouseWheelUp Method

OnMouseWheelUp event dispatcher.

Pascal

```
function DoMouseWheelUp(Shift: TShiftState; MousePos: TPoint): Boolean; override;
```

C++

```
virtual __fastcall Boolean DoMouseWheelUp(TShiftState Shift, TPoint MousePos);
```

Description

The DoMouseWheel method calls DoMouseWheelUp to generate an OnMouseWheelUp event.

The Shift parameter indicates the state of the Shift, Alt, and Ctrl keys.

The MousePos parameter indicates the position of the mouse pointer.

DoMouseWheelUp returns true if an event handler indicates that it handled the message, and false otherwise.

Override DoMouseWheelUp to perform class-specific tasks in addition to calling the event handler when the mouse wheel is rotated upward.

See Also

OnMouseWheelUp

DoMouseWheel

62.1.4.5.31 TCustomRibbon.DoRecentItemClick Method

Pascal

```
procedure DoRecentItemClick(const FileName: string; const Index: Integer); virtual;
```

C++

```
virtual __fastcall DoRecentItemClick(const UnicodeString FileName, const int Index);
```

Description

This is DoRecentItemClick, a member of class TCustomRibbon.

62.1.4.5.32 TCustomRibbon.DoScrollTabsLeft Method

Pascal

```
procedure DoScrollTabsLeft; virtual;
```

C++

```
virtual __fastcall DoScrollTabsLeft();
```

Description

This is DoScrollTabsLeft, a member of class TCustomRibbon.

62.1.4.5.33 TCustomRibbon.DoScrollTabsRight Method

Pascal

```
procedure DoScrollTabsRight; virtual;
```

C++

```
virtual __fastcall DoScrollTabsRight();
```

Description

This is DoScrollTabsRight, a member of class TCustomRibbon.

62.1.4.5.34 TCustomRibbon.DoTabChange Method

Pascal

```
procedure DoTabChange(const NewIndex: Integer; const OldIndex: Integer; var AllowChange:
Boolean); virtual;
```

C++

```
virtual __fastcall DoTabChange(const int NewIndex, const int OldIndex, Boolean AllowChange);
```

Description

This is DoTabChange, a member of class TCustomRibbon.

62.1.4.5.35 TCustomRibbon.DrawCaption Method

Pascal

```
procedure DrawCaption; virtual;
```

C++

```
virtual __fastcall DrawCaption();
```

Description

This is DrawCaption, a member of class TCustomRibbon.

62.1.4.5.36 TCustomRibbon.DrawQuickAccessToolbar Method

Pascal

```
procedure DrawQuickAccessToolbar; virtual;
```

C++

```
virtual __fastcall DrawQuickAccessToolbar();
```

Description

This is DrawQuickAccessToolbar, a member of class TCustomRibbon.

62.1.4.5.37 TCustomRibbon.DrawSystemButtons Method

Pascal

```
procedure DrawSystemButtons;
```

C++

```
__fastcall DrawSystemButtons();
```

Description

This is DrawSystemButtons, a member of class TCustomRibbon.

62.1.4.5.38 TCustomRibbon.DrawTab Method

Pascal

```
procedure DrawTab(const Index: Integer); virtual;
```

C++

```
virtual __fastcall DrawTab(const int Index);
```

Description

This is DrawTab, a member of class TCustomRibbon.

62.1.4.5.39 TCustomRibbon.EndUpdate Method

Pascal

```
procedure EndUpdate;
```

C++

```
__fastcall EndUpdate();
```

Description

This is EndUpdate, a member of class TCustomRibbon.

62.1.4.5.40 TCustomRibbon.GetContextMenuOptions Method

Pascal

```
function GetContextMenuOptions: TContextMenuOptions;
```

C++

```
__fastcall TContextMenuOptions GetContextMenuOptions();
```

Description

This is GetContextMenuOptions, a member of class TCustomRibbon.

62.1.4.5.41 TCustomRibbon.GetHelpButtonRect Method

Pascal

```
function GetHelpButtonRect: TRect; virtual;
```

C++

```
virtual __fastcall TRect GetHelpButtonRect();
```

Description

This is GetHelpButtonRect, a member of class TCustomRibbon.

62.1.4.5.42 TCustomRibbon.GetIcon Method

Return Icon used by Application Menu (see page 3397) Button.

Pascal

```
function GetIcon(out IconSize: TIconSize): TIcon; virtual;
```

C++

```
virtual __fastcall TIcon GetIcon(TIconSize IconSize);
```

Description

GetIcon returns the Icon used by the Application Menu (see page 3397) Button. If there is no Icon assigned to the TApplicationMenu.Icon property, the application's icon is used.

See Also

TApplicationMenu (see page 3350)

Icon (see page 3354)

62.1.4.5.43 TCustomRibbon.GetRibbonHeight Method

Pascal

```
function GetRibbonHeight: Integer;
```

C++

```
__fastcall int GetRibbonHeight();
```

Description

This is GetRibbonHeight, a member of class TCustomRibbon.

62.1.4.5.44 TCustomRibbon.GetRibbonMetric Method

Pascal

```
function GetRibbonMetric(const Metric: TRibbonMetric): Integer;
```

C++

```
__fastcall int GetRibbonMetric(const TRibbonMetric Metric);
```

Description

This is GetRibbonMetric, a member of class TCustomRibbon.

62.1.4.5.45 TCustomRibbon.GetTabAt Method

Pascal

```
function GetTabAt(const X: Integer; const Y: Integer): Integer; virtual;
```

C++

```
virtual __fastcall int GetTabAt(const int X, const int Y);
```

Description

This is GetTabAt, a member of class TCustomRibbon.

62.1.4.5.46 TCustomRibbon.GetTabRect Method

Pascal

```
function GetTabRect(const Index: Integer): TRect; virtual;
```

C++

```
virtual __fastcall TRect GetTabRect(const int Index);
```

Description

This is GetTabRect, a member of class TCustomRibbon.

62.1.4.5.47 TCustomRibbon.GetTabWidth Method

Pascal

```
function GetTabWidth(const Index: Integer): Integer; virtual;
```

C++

```
virtual __fastcall int GetTabWidth(const int Index);
```

Description

This is GetTabWidth, a member of class TCustomRibbon.

62.1.4.5.48 TCustomRibbon.HasApplicationMenu Method

Pascal

```
function HasApplicationMenu: Boolean;
```

C++

```
__fastcall Boolean HasApplicationMenu();
```

Description

This is HasApplicationMenu, a member of class TCustomRibbon.

62.1.4.5.49 TCustomRibbon.HasQuickAccessToolbar Method

Pascal

```
function HasQuickAccessToolbar: Boolean;
```

C++

```
__fastcall Boolean HasQuickAccessToolbar();
```

Description

This is HasQuickAccessToolbar, a member of class TCustomRibbon.

62.1.4.5.50 TCustomRibbon.Idle Method

Pascal

```
procedure Idle(const Msg: TMsg);
```

C++

```
__fastcall Idle(const TMsg Msg);
```

Description

This is Idle, a member of class TCustomRibbon.

62.1.4.5.51 TCustomRibbon.InvalidateHeaderArea Method

Invalidate header section of Ribbon (see page 3347).

Pascal

```
procedure InvalidateHeaderArea;
```

C++

```
__fastcall InvalidateHeaderArea();
```

Description

InvalidateHeaderArea invalidates the header section of the Ribbon (see page 3347). Invalidating the header of the ribbon is especially important when using a custom frame. When using a custom frame, the header section of the ribbon needs to react visually to additional non-client specific messages.

See Also

InvalidateTabArea (see page 3383)

62.1.4.5.52 TCustomRibbon.InvalidateScrollButtons Method

Pascal

```
procedure InvalidateScrollButtons;
```

C++

```
__fastcall InvalidateScrollButtons();
```

Description

This is InvalidateScrollButtons, a member of class TCustomRibbon.

62.1.4.5.53 TCustomRibbon.InvalidateSystemMenuArea Method

Pascal

```
procedure InvalidateSystemMenuArea;
```

C++

```
__fastcall InvalidateSystemMenuArea();
```

Description

This is InvalidateSystemMenuArea, a member of class TCustomRibbon.

62.1.4.5.54 TCustomRibbon.InvalidateTabArea Method

Invalidate area that displays tabs.

Pascal

```
procedure InvalidateTabArea;
```

C++

```
__fastcall InvalidateTabArea();
```

Description

InvalidateTabArea invalidates the area that displays the tabs.

See Also

InvalidateHeaderArea (see page 3382)

62.1.4.5.55 TCustomRibbon.KeyDown Method

Respond to key press events.

Pascal

```
procedure KeyDown(var Key: Word; Shift: TShiftState); override;
```

C++

```
virtual __fastcall KeyDown(Word Key, TShiftState Shift);
```

Description

When a windowed control receives a key-down message (WM_KEYDOWN) from Windows, its message handler calls the DoKeyDown method. If DoKeyDown determines that the control should, in fact, process the character, it calls KeyDown, passing the key code and shift-key state in the Key and Shift parameters, respectively.

KeyDown calls any event handler attached to the OnKeyDown event. Override KeyDown to provide other responses in addition to the event-handler call.

The Key parameter is the key on the keyboard. For non-alphanumeric keys, you must use WinAPI virtual key codes to determine the key pressed. For more information, search for virtual key codes in the Win32 Developer's Reference (WIN32.HLP).

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

Either KeyDown or the OnKeyDown event handler it calls can suppress further processing of a key by setting the Key parameter to zero.

See Also

KeyUp

OnKeyDown

DoKeyDown

DoKeyPress

DoKeyUp

62.1.4.5.56 TCustomRibbon.Loaded Method

Initializes the control after it is loaded from a stream.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

Do not call Loaded. The VCL streaming system calls Loaded automatically after the control's form is loaded into memory so that the control can complete any initializations that depend on other objects in the form.

Loaded overrides the inherited method in order to initialize the control from its associated Action. To change the properties the control copies from its action, override the ActionChange method.

See Also

Loaded

Action

ActionChange

62.1.4.5.57 TCustomRibbon.MouseDown Method

OnMouseDown event dispatcher.

Pascal

```
procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer);
override;
```

C++

```
virtual __fastcall MouseDown(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user presses the mouse button down while the cursor's hotspot is over the control.

The Button parameter determines which mouse button the user pressed. Shift indicates which shift keys (Shift, Ctrl, or Alt) were down when the user pressed the mouse button. X and Y are the pixel coordinates of the mouse pointer within the client area of the control.

A control calls MouseDown in response to any of the Windows mouse-down messages (WM_LBUTTONDOWN, WM_MBUTTONDOWN, WM_RBUTTONDOWN), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was

pressed down: left, right, or middle.

See Also

MouseMove

MouseUp

OnMouseDown

62.1.4.5.58 TCustomRibbon.MouseInSystemMenu Method

Pascal

```
function MouseInSystemMenu(const LPt: TPoint): Boolean;
```

C++

```
__fastcall Boolean MouseInSystemMenu(const TPoint LPt);
```

Description

This is MouseInSystemMenu, a member of class TCustomRibbon.

62.1.4.5.59 TCustomRibbon.MouseMove Method

Respond to mouse moving over control area..

Pascal

```
procedure MouseMove(Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseMove(TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseMove method to provide other responses in addition to calling the OnMouseMove event handler when the user moves the mouse.

A control calls MouseMove in response to any of the Windows mouse-move messages (WM_MOUSEMOVE), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively.

As the mouse cursor moves across a control, this method is called repeatedly. Each time it is called, it is with the new coordinates that reflect the continuous path of the mouse cursor across the screen real estate covered by the control's visual representation.

See Also

MouseDown

MouseUp

OnMouseMove

62.1.4.5.60 TCustomRibbon.MouseUp Method

OnMouseUp event dispatcher.

Pascal

```
procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseUp(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user releases a previously pressed mouse button while the cursor's hotspot is over the control.

A control calls MouseUp in response to any of the Windows mouse-up messages (WM_LBUTTONUP, WM_MBUTTONUP, WM_RBUTTONUP), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was released: left, right, or middle

See Also

MouseDown

MouseMove

OnMouseUp

62.1.4.5.61 TCustomRibbon.MoveGroup Method

Move group to new page.

Pascal

```
procedure MoveGroup(const GroupName: string; const PageName: string);
```

C++

```
__fastcall MoveGroup(const UnicodeString GroupName, const UnicodeString PageName);
```

Description

MoveGroup moves a group to a new page. The group to be moved is specified in the **GroupName** parameter. The destination page is specified in the **PageName** parameter.

62.1.4.5.62 TCustomRibbon.Notification Method

Responds to notifications that components are being created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification allows TControl to update its PopupMenu property when the pop-up menu it refers to is destroyed.

Override the Notification method to respond to notifications that other components are about to be destroyed or have just been created. Use the Notification method to update controls that rely on other objects. Remove references to objects that are going away so that the control does not refer to objects that have been destroyed. For example, data-aware objects override the Notification method to update themselves when their data source is removed. Some objects respond to notifications that other objects of a specific type are being created. For example, the session component sets the Session property of new data-aware controls to itself when the AutoSessionName property is true.

By default, components pass along the notification to their owned components, if any.

See Also

PopupMenu

Notification

62.1.4.5.63 TCustomRibbon.Paint Method

Renders the image of a custom control.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method for TCustomControl does nothing other than provide the interface for a method that responds to WM_PAINT messages. Paint is called by the PaintWindow method, after that method has supplied the Canvas with the handle to a device context. When creating a custom control, always override Paint to draw the image of the control.

Tip: To determine which portions of the control's canvas need to be repainted when Paint is called, use the ClipRect property of the canvas.

See Also

Canvas

PaintWindow

GetDeviceContext

ClipRect

62.1.4.5.64 TCustomRibbon.RebuildRibbonMetrics Method

Pascal

```
procedure RebuildRibbonMetrics;
```

C++

```
__fastcall RebuildRibbonMetrics();
```

Description

This is RebuildRibbonMetrics, a member of class TCustomRibbon.

62.1.4.5.65 TCustomRibbon.RemoveAllTabs Method

Remove all Tabs (see page 3399) from Ribbon (see page 3347).

Pascal

```
procedure RemoveAllTabs;
```

C++

```
__fastcall RemoveAllTabs();
```

Description

RemoveAllTabs removes all Tabs (see page 3399) from the Ribbon (see page 3347). All associated pages and groups are deleted.

See Also

RemoveTab (see page 3388)

62.1.4.5.66 TCustomRibbon.RemoveRecentItem Method

Pascal

```
function RemoveRecentItem(const FileName: string): Boolean;
```

C++

```
__fastcall Boolean RemoveRecentItem(const UnicodeString FileName);
```

Description

This is RemoveRecentItem, a member of class TCustomRibbon.

62.1.4.5.67 TCustomRibbon.RemoveTab Method

Remove tab with specified caption.

Pascal

```
procedure RemoveTab(const ACaption: string);
```

C++

```
__fastcall RemoveTab(const UnicodeString ACaption);
```

Description

RemoveTab removes the tab with the specified caption. RemoveTab deletes the associated page and any groups on that page.

See Also

RemoveAllTabs (see page 3387)

62.1.4.5.68 TCustomRibbon.Resize Method

Respond to control resize.

Pascal

```
procedure Resize; override;
```

C++

```
virtual __fastcall Resize();
```

Description

Resize is called automatically immediately after the control's dimensions change. As implemented in TControl, Resize calls the control's OnResize event handler if it is assigned.

See Also

OnResize

62.1.4.5.69 TCustomRibbon.SetBounds Method

Sets the windowed control's boundary properties all at once.

Pascal

```
procedure SetBounds(ALeft: Integer; ATop: Integer; AWidth: Integer; AHeight: Integer);
override;
```

C++

```
virtual __fastcall SetBounds(int ALeft, int ATop, int AWidth, int AHeight);
```

Description

Use SetBounds to change all of the control's boundary properties at once. The same effect can be achieved by setting the Left, Top, Width, and Height properties separately. By setting all four properties at once, SetBounds ensures that the control will not repaint between changes

Specify the values for the Left, Top, Width, and Height properties as the value of the ALeft, ATop, AWidth, and AHeight parameters, respectively.

See Also

Height

Left

Top

Width

62.1.4.5.70 TCustomRibbon.SetParent Method

Sets the parent of the control.

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

SetParent is the protected implementation of the Parent property. Override SetParent to execute additional code when the value of the Parent property changes.

The AParent parameter specifies the new parent of the control.

If the control already has a parent, SetParent removes the control from that parent's list of controls by calling the parent's RemoveControl method. If AParent is not nil (Delphi) or NULL (C++), SetParent calls its InsertControl method to add the control to its list of controls.

The TControl destructor calls SetParent(nil (Delphi) or NULL (C++)) to remove the control from its parent's control list before destroying the component.

See Also

Destroy

InsertControl

Parent

RemoveControl

62.1.4.5.71 TCustomRibbon.TabHandledKeyTips Method

Validate **KeyTip** input when **KeyTip**s State is **ktsTabs**.

Pascal

```
procedure TabHandledKeyTips(const KeyTip: string; var KeyTips: TKeyTipList);
```

C++

```
__fastcall TabHandledKeyTips(const UnicodeString KeyTip, TKeyTipList KeyTips);
```

Description

TabHandledKeyTips validates **KeyTip** input when the **KeyTip**s State is **ktsTabs**. **KeyTip**s can be in either **ktsTabs** or **ktsControls** state. When the **KeyTip**s state is **ktsTabs**, the user has the option of selecting which Tab to switch to and display the **KeyTip**s for the controls on that tab. When the State **ktsTabs**, the Application Button and Quick Access Toolbar are also displaying their **KeyTip**s.

The parameters are:

- KeyTip: The **KeyTip** the user has input that needs to be validated against.
- Full: A list of TKeyTipMatchItem (see page 3598) that contain details if any **KeyTip**s were matched.

See Also

TKeyTipMatchItem (see page 3598)

TKeyTipState (see page 3928)

62.1.4.5.72 TCustomRibbon.TabHideKeyTips Method

Hide K**eyTip**s.

Pascal

```
procedure TabHideKeyTips(const KeyTip: string; KeyTips: TKeyTipList);
```

C++

```
__fastcall TabHideKeyTips(const UnicodeString KeyTip, TKeyTipList KeyTips);
```

Description

TabHideKeyTips hides the **KeyTip**s. If the **KeyTips** parameter is `nil`, all **KeyTip**s are hidden. Otherwise **KeyTips** contains TKeyTipMatchItem (see page 3598) that are not supposed to be hidden at this time. This would occur when there are a number of Partial **KeyTip** matches where the first character of a **KeyTip** matches many **KeyTip**s.

See Also

TKeyTipMatchItem (see page 3598)

TabShowKeyTips (see page 3390)

62.1.4.5.73 TCustomRibbon.TabShowKeyTips Method

Display **KeyTip**s for **ktsTabs** State.

Pascal

```
procedure TabShowKeyTips;
```

C++

```
__fastcall TabShowKeyTips();
```

Description

TabShowKeyTips displays the **KeyTip**s for the **ktsTabs** State. When called, **KeyTip**s are displayed for the Tabs (see page 3399), Application Button and Quick Access Toolbar.

See Also

TabHideKeyTips (see page 3390)

TKeyTipState (see page 3928)

62.1.4.5.74 TCustomRibbon.UMCheckResize Method

Pascal

```
procedure UMCheckResize(var Message: TMessage); message UM_CHECKSIZE;
```

C++

```
__fastcall UMCheckResize(TMessage Message);
```

Description

This is UMCheckResize, a member of class TCustomRibbon.

62.1.4.5.75 TCustomRibbon.UMDisplayKeyTips Method

Pascal

```
procedure UMDisplayKeyTips(var Message: TMessage); message UM_DISPLAYKEYTIPS;
```

C++

```
__fastcall UMDisplayKeyTips(TMessage Message);
```

Description

This is UMDisplayKeyTips, a member of class TCustomRibbon.

62.1.4.5.76 TCustomRibbon.UpdateQATPosition Method

Pascal

```
procedure UpdateQATPosition;
```

C++

```
__fastcall UpdateQATPosition();
```

Description

This is UpdateQATPosition, a member of class TCustomRibbon.

62.1.4.5.77 TCustomRibbon.UpdateSystemMenuInfo Method

Pascal

```
procedure UpdateSystemMenuInfo;
```

C++

```
__fastcall UpdateSystemMenuInfo();
```

Description

This is UpdateSystemMenuInfo, a member of class TCustomRibbon.

62.1.4.5.78 TCustomRibbon.ValidTab Method

Pascal

```
function ValidTab(const Index: Integer): Boolean;
```

C++

```
__fastcall Boolean ValidTab(const int Index);
```

Description

This is ValidTab, a member of class TCustomRibbon.

62.1.4.5.79 TCustomRibbon.WMEraseBkgnd Method

Pascal

```
procedure WMEraseBkgnd(var Message: TWMEraseBkgnd); message WM_ERASEBKGND;
```

C++

```
__fastcall WMEraseBkgnd(TWMEraseBkgnd Message);
```

Description

This is WMEraseBkgnd, a member of class TCustomRibbon.

62.1.4.5.80 TCustomRibbon.WMGetDlgCode Method

Pascal

```
procedure WMGetDlgCode(var Message: TWMGetDlgCode); message WM_GETDLGCODE;
```

C++

```
__fastcall WMGetDlgCode(TWMGetDlgCode Message);
```

Description

This is WMGetDlgCode, a member of class TCustomRibbon.

62.1.4.5.81 TCustomRibbon.WMKillFocus Method

Pascal

```
procedure WMKillFocus(var Message: TWMKillFocus); message WM_KILLFOCUS;
```

C++

```
__fastcall WMKillFocus(TWMKillFocus Message);
```

Description

This is WMKillFocus, a member of class TCustomRibbon.

62.1.4.5.82 TCustomRibbon.WMSettingChange Method

Pascal

```
procedure WMSettingChange(var Message: TWMSettingChange); message WM_SETTINGCHANGE;
```

C++

```
__fastcall WMSettingChange(TWMSettingChange Message);
```

Description

This is WMSettingChange, a member of class TCustomRibbon.

62.1.4.5.83 TCustomRibbon.WMWindowPosChanged Method

Pascal

```
procedure WMWindowPosChanged(var Message: TWMWindowPosChanged); message WM_WINDOWPOSCHANGED;
```

C++

```
__fastcall WMWindowPosChanged(TWMWindowPosChanged Message);
```

Description

This is WMWindowPosChanged, a member of class TCustomRibbon.

62.1.4.6 TCustomRibbon Properties

The properties of the TCustomRibbon class are listed here.

Protected Properties

	Name	Description
	AllowKeyboardNavigation (see page 3395)	This is AllowKeyboardNavigation, a member of class TCustomRibbon.
	HideRibbon (see page 3396)	This is HideRibbon, a member of class TCustomRibbon.
	HotTabIndex (see page 3396)	This is HotTabIndex, a member of class TCustomRibbon.
	KeyTipsVisible (see page 3397)	This is KeyTipsVisible, a member of class TCustomRibbon.
R	Menu (see page 3397)	This is Menu, a member of class TCustomRibbon.
R	ScaleFactor (see page 3398)	This is ScaleFactor, a member of class TCustomRibbon.

Public Properties

	Name	Description
	ActionManager (see page 3394)	This is ActionManager, a member of class TCustomRibbon.
	Active (see page 3394)	Indicates if form that ribbon is placed on is active form.
	ActivePage (see page 3394)	Page currently displayed by ribbon control.
	ApplicationMenu (see page 3395)	Change appearance of ribbon's associated TRibbonApplicationMenuBar (see page 3942) control.
	ColorMap (see page 3395)	This is ColorMap, a member of class TCustomRibbon.
	DocumentName (see page 3396)	Text in Ribbon (see page 3347) caption.
	HideTabs (see page 3396)	Shows or hides Tab portion of Ribbon (see page 3347).
	Minimized (see page 3397)	Show Ribbon (see page 3347) in minimized state where tabs can be selected but are not visible.
	QuickAccessToolbar (see page 3397)	Properties associated with TQuickAccessToolbar (see page 3469).
	ScreenTips (see page 3398)	This is ScreenTips, a member of class TCustomRibbon.
	ShowHelpButton (see page 3398)	Determines whether ribbon displays Help Button.
	Style (see page 3398)	This is Style, a member of class TCustomRibbon.
	TabIndex (see page 3399)	Change the active page of ribbon.

	Tabs (see page 3399)	Collection of TRibbonTabs (see page 3560) visible on ribbon.
	UseCustomFrame (see page 3399)	Indicates whether to use a custom frame for ribbon controls.

62.1.4.6.1 TCustomRibbon.ActionManager Property

Pascal

```
property ActionManager: TCustomActionManager;
```

C++

```
__property TCustomActionManager ActionManager;
```

Description

This is ActionManager, a member of class TCustomRibbon.

62.1.4.6.2 TCustomRibbon.Active Property

Indicates if form that ribbon is placed on is active form.

Pascal

```
property Active: Boolean;
```

C++

```
__property Boolean Active;
```

Description

Active indicates if the form that the ribbon is placed on is the active form of the desktop.

There should be no need for a user to set the Active property unless they are not using the custom frame functionality of the ribbon. The appearance of the ribbon changes depending on the value of this property.

If the form that the ribbon is placed on does not descend from TRibbonForm, the user should update the Active property when the form gets activated and deactivated.

See Also

ActivePage (see page 3394)

62.1.4.6.3 TCustomRibbon.ActivePage Property

Page currently displayed by ribbon control.

Pascal

```
property ActivePage: TCustomRibbonPage;
```

C++

```
__property TCustomRibbonPage ActivePage;
```

Description

ActivePage is the page currently displayed by the ribbon control.

Read ActivePage to determine which page is selected by the user. The user selects a page by clicking the mouse on its tab or selects the page's associated keytip. Set ActivePage to bring a page to the foreground. Only one page can be active at a time.

Use TRibbonTabs.SelectNext and TRibbonTabs.SelectPrior to select the next and prior pages in the ribbon. If the active page does not change after a call to SelectNext or SelectPrior, the return value of those functions is False.

To change the active page of the ribbon by Index, set the TabIndex (see page 3399) property to the desired page index.

See Also

TRibbonTabs (see page 3560)

Active (see page 3394)

TabIndex (see page 3399)

SelectNext (see page 3563)

SelectPrior (see page 3563)

62.1.4.6.4 TCustomRibbon.AllowKeyboardNavigation Property

Pascal

```
property AllowKeyboardNavigation: Boolean;
```

C++

```
__property Boolean AllowKeyboardNavigation;
```

Description

This is AllowKeyboardNavigation, a member of class TCustomRibbon.

62.1.4.6.5 TCustomRibbon.ApplicationMenu Property

Change appearance of ribbon's associated TRibbonApplicationMenuBar (see page 3942) control.

Pascal

```
property ApplicationMenu: TApplicationMenu;
```

C++

```
__property TApplicationMenu ApplicationMenu;
```

Description

ApplicationMenu changes the appearance for the ribbon's associated TRibbonApplicationMenuBar (see page 3942) control.

Changing the values of the fields in ApplicationMenu changes the characteristics of the Application Menu (see page 3397) for the ribbon. The Application Menu (see page 3397) on the ribbon is represented by the round button in the top left corner.

See Also

TApplicationMenu (see page 3350)

TRibbonApplicationMenuBar (see page 3942)

62.1.4.6.6 TCustomRibbon.ColorMap Property

Pascal

```
property ColorMap: TCustomActionBarColorMap;
```

C++

```
__property TCustomActionBarColorMap ColorMap;
```

Description

This is ColorMap, a member of class TCustomRibbon.

62.1.4.6.7 TCustomRibbon.DocumentName Property

Text in Ribbon (see page 3347) caption.

Pascal

```
property DocumentName: string;
```

C++

```
__property UnicodeString DocumentName;
```

Description

DocumentName appears in the Ribbon (see page 3347) caption.

62.1.4.6.8 TCustomRibbon.HideRibbon Property

Pascal

```
property HideRibbon: Boolean;
```

C++

```
__property Boolean HideRibbon;
```

Description

This is HideRibbon, a member of class TCustomRibbon.

62.1.4.6.9 TCustomRibbon.HideTabs Property

Shows or hides Tab portion of Ribbon (see page 3347).

Pascal

```
property HideTabs: Boolean;
```

C++

```
__property Boolean HideTabs;
```

Description

HideTabs shows or hides the Tab portion of the Ribbon (see page 3347). When HideTabs is True, the user cannot access any of the Tabs (see page 3399) on the Ribbon (see page 3347).

62.1.4.6.10 TCustomRibbon.HotTabIndex Property

Pascal

```
property HotTabIndex: Integer;
```

C++

```
__property int HotTabIndex;
```

Description

This is HotTabIndex, a member of class TCustomRibbon.

62.1.4.6.11 TCustomRibbon.KeyTipsVisible Property

Pascal

```
property KeyTipsVisible: Boolean;
```

C++

```
__property Boolean KeyTipsVisible;
```

Description

This is KeyTipsVisible, a member of class TCustomRibbon.

62.1.4.6.12 TCustomRibbon.Menu Property

Pascal

```
property Menu: TRibbonApplicationMenuBar;
```

C++

```
__property TRibbonApplicationMenuBar Menu;
```

Description

This is Menu, a member of class TCustomRibbon.

62.1.4.6.13 TCustomRibbon.Minimized Property

Show Ribbon (see page 3347) in minimized state where tabs can be selected but are not visible.

Pascal

```
property Minimized: Boolean;
```

C++

```
__property Boolean Minimized;
```

Description

Minimized indicates whether to show the Ribbon (see page 3347) in a minimized state where tabs can be selected but are not visible. When Minimized is True, Tabs (see page 3399) still function. You can access the tabs by selecting the desired Tab with the mouse or switching to the tab by using tab allocated `KeyTips`.

62.1.4.6.14 TCustomRibbon.QuickAccessToolbar Property

Properties associated with TQuickAccessToolbar (see page 3469).

Pascal

```
property QuickAccessToolbar: TQuickAccessToolbar;
```

C++

```
__property TQuickAccessToolbar QuickAccessToolbar;
```

Description

QuickAccessToolbar has the properties associated with the TQuickAccessToolbar (see page 3469) on the ribbon. Use these properties to change the appearance of the Quick Access Toolbar as displayed on the ribbon.

See Also

TQuickAccessToolbar (see page 3469)

62.1.4.6.15 TCustomRibbon.ScaleFactor Property

Pascal

```
property ScaleFactor: Single;
```

C++

```
__property Single ScaleFactor;
```

Description

This is ScaleFactor, a member of class TCustomRibbon.

62.1.4.6.16 TCustomRibbon.ScreenTips Property

Pascal

```
property ScreenTips: TCustomScreenTipsManager;
```

C++

```
__property TCustomScreenTipsManager ScreenTips;
```

Description

This is ScreenTips, a member of class TCustomRibbon.

62.1.4.6.17 TCustomRibbon.ShowHelpButton Property

Determines whether ribbon displays Help Button.

Pascal

```
property ShowHelpButton: Boolean;
```

C++

```
__property Boolean ShowHelpButton;
```

Description

ShowHelpButton indicates whether or not the ribbon displays a Help Button in the top right hand corner.

See Also

THelpButton (see page 3459)

62.1.4.6.18 TCustomRibbon.Style Property

Pascal

```
property Style: TRibbonStyleActionBars;
```

C++

```
__property TRibbonStyleActionBars Style;
```

Description

This is Style, a member of class TCustomRibbon.

62.1.4.6.19 TCustomRibbon.TabIndex Property

Change the active page of ribbon.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

TabIndex changes the active page of the ribbon. Set the TabIndex property to the desired page index.

See Also

Tabs (see page 3399)

62.1.4.6.20 TCustomRibbon.Tabs Property

Collection of TRibbonTabs (see page 3560) visible on ribbon.

Pascal

```
property Tabs: TRibbonTabs;
```

C++

```
__property TRibbonTabs Tabs;
```

Description

Tabs contains a collection of TRibbonTabs (see page 3560) visible on the ribbon.

Tabs is a linked list of TRibbonTabs (see page 3560) objects. TRibbonTabs (see page 3560) is a simple data structure that contains the caption for the tab, the keytip for the tab as well as a reference to the Tab object.

See Also

TRibbonTabs (see page 3560)

TabIndex (see page 3399)

62.1.4.6.21 TCustomRibbon.UseCustomFrame Property

Indicates whether to use a custom frame for ribbon controls.

Pascal

```
property UseCustomFrame: Boolean;
```

C++

```
__property Boolean UseCustomFrame;
```

Description

UseCustomFrame indicates whether or not to use a custom frame for ribbon controls.

62.1.4.7 TCustomRibbon Nested Types

The nested types of the TCustomRibbon class are listed here.

Public Nested Types

Name	Description
TTabChangeEvent (see page 3400)	This is nested type Ribbon.TCustomRibbon.TTabChangeEvent.

62.1.4.7.1 TCustomRibbon.TTabChangeEvent Nested Type

Pascal

```
TTabChangeEvent = procedure (Sender: TObject; const NewIndex, OldIndex: Integer; var
AllowChange: Boolean) of object;
```

C++

```
(Sender: TObject; const NewIndex, OldIndex: Integer; var AllowChange: Boolean) (
TTabChangeEvent)();
```

Description

This is nested type Ribbon.TCustomRibbon.TTabChangeEvent.

62.1.4.8 TCustomRibbon Events

The events of the TCustomRibbon class are listed here.

Protected Events

	Name	Description
	OnLeftTabScrollButtonClick (see page 3400)	This is OnLeftTabScrollButtonClick, a member of class TCustomRibbon.
	OnRightTabScrollButtonClick (see page 3401)	This is OnRightTabScrollButtonClick, a member of class TCustomRibbon.

Public Events

	Name	Description
	OnHelpButtonClick (see page 3400)	This is OnHelpButtonClick, a member of class TCustomRibbon.
	OnRecentItemClick (see page 3401)	This is OnRecentItemClick, a member of class TCustomRibbon.
	OnTabChange (see page 3401)	This is OnTabChange, a member of class TCustomRibbon.

62.1.4.8.1 TCustomRibbon.OnHelpButtonClick Event

Pascal

```
property OnHelpButtonClick: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnHelpButtonClick;
```

Description

This is OnHelpButtonClick, a member of class TCustomRibbon.

62.1.4.8.2 TCustomRibbon.OnLeftTabScrollButtonClick Event

Pascal

```
property OnLeftTabScrollButtonClick: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnLeftTabScrollButtonClick;
```

Description

This is OnLeftTabScrollButtonClick, a member of class TCustomRibbon.

62.1.4.8.3 TCustomRibbon.OnRecentItemClick Event

Pascal

```
property OnRecentItemClick: TRecentItemClickEvent;
```

C++

```
__property TRecentItemClickEvent OnRecentItemClick;
```

Description

This is OnRecentItemClick, a member of class TCustomRibbon.

62.1.4.8.4 TCustomRibbon.OnRightTabScrollButtonClick Event

Pascal

```
property OnRightTabScrollButtonClick: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnRightTabScrollButtonClick;
```

Description

This is OnRightTabScrollButtonClick, a member of class TCustomRibbon.

62.1.4.8.5 TCustomRibbon.OnTabChange Event

Pascal

```
property OnTabChange: TTabChangeEvent;
```

C++

```
__property TTabChangeEvent OnTabChange;
```

Description

This is OnTabChange, a member of class TCustomRibbon.

62.1.5 TCustomRibbonColorMap Class

Class Hierarchy

Pascal

```
TCustomRibbonColorMap = class(TXPColorMap);
```

C++

```
class TCustomRibbonColorMap : public TXPColorMap;
```

File

Ribbon

Description

This is class Ribbon.TCustomRibbonColorMap.

62.1.5.1 TCustomRibbonColorMap Members

The following tables list the members exposed by TCustomRibbonColorMap.

Public Properties

	Name	Description
	ActiveTabFontColor (see page 3402)	This is ActiveTabFontColor, a member of class TCustomRibbonColorMap.
	BevelMinimizedInner (see page 3403)	This is BevelMinimizedInner, a member of class TCustomRibbonColorMap.
	BevelMinimizedOuter (see page 3403)	This is BevelMinimizedOuter, a member of class TCustomRibbonColorMap.
	CaptionFontColor (see page 3403)	This is CaptionFontColor, a member of class TCustomRibbonColorMap.
	ControlColor (see page 3403)	This is ControlColor, a member of class TCustomRibbonColorMap.
	ControlFrameColor (see page 3404)	This is ControlFrameColor, a member of class TCustomRibbonColorMap.
	ControlInactiveColor (see page 3404)	This is ControlInactiveColor, a member of class TCustomRibbonColorMap.
	DocumentFontColor (see page 3404)	This is DocumentFontColor, a member of class TCustomRibbonColorMap.
	GroupFontColor (see page 3404)	This is GroupFontColor, a member of class TCustomRibbonColorMap.
	InactiveTabFontColor (see page 3404)	This is InactiveTabFontColor, a member of class TCustomRibbonColorMap.
	QuickAccessToolbarMoreBorderColor (see page 3405)	This is QuickAccessToolbarMoreBorderColor, a member of class TCustomRibbonColorMap.
	QuickAccessToolbarMoreColor (see page 3405)	This is QuickAccessToolbarMoreColor, a member of class TCustomRibbonColorMap.

62.1.5.2 TCustomRibbonColorMap Properties

The properties of the TCustomRibbonColorMap class are listed here.

Public Properties

	Name	Description
	ActiveTabFontColor (see page 3402)	This is ActiveTabFontColor, a member of class TCustomRibbonColorMap.
	BevelMinimizedInner (see page 3403)	This is BevelMinimizedInner, a member of class TCustomRibbonColorMap.
	BevelMinimizedOuter (see page 3403)	This is BevelMinimizedOuter, a member of class TCustomRibbonColorMap.
	CaptionFontColor (see page 3403)	This is CaptionFontColor, a member of class TCustomRibbonColorMap.
	ControlColor (see page 3403)	This is ControlColor, a member of class TCustomRibbonColorMap.
	ControlFrameColor (see page 3404)	This is ControlFrameColor, a member of class TCustomRibbonColorMap.
	ControlInactiveColor (see page 3404)	This is ControlInactiveColor, a member of class TCustomRibbonColorMap.
	DocumentFontColor (see page 3404)	This is DocumentFontColor, a member of class TCustomRibbonColorMap.
	GroupFontColor (see page 3404)	This is GroupFontColor, a member of class TCustomRibbonColorMap.
	InactiveTabFontColor (see page 3404)	This is InactiveTabFontColor, a member of class TCustomRibbonColorMap.
	QuickAccessToolbarMoreBorderColor (see page 3405)	This is QuickAccessToolbarMoreBorderColor, a member of class TCustomRibbonColorMap.
	QuickAccessToolbarMoreColor (see page 3405)	This is QuickAccessToolbarMoreColor, a member of class TCustomRibbonColorMap.

62.1.5.2.1 TCustomRibbonColorMap.ActiveTabFontColor Property

Pascal

```
property ActiveTabFontColor: TColor;
```

C++

```
__property TColor ActiveTabFontColor;
```

Description

This is ActiveTabFontColor, a member of class TCustomRibbonColorMap.

62.1.5.2.2 TCustomRibbonColorMap.BevelMinimizedInner Property

Pascal

```
property BevelMinimizedInner: TColor;
```

C++

```
__property TColor BevelMinimizedInner;
```

Description

This is BevelMinimizedInner, a member of class TCustomRibbonColorMap.

62.1.5.2.3 TCustomRibbonColorMap.BevelMinimizedOuter Property

Pascal

```
property BevelMinimizedOuter: TColor;
```

C++

```
__property TColor BevelMinimizedOuter;
```

Description

This is BevelMinimizedOuter, a member of class TCustomRibbonColorMap.

62.1.5.2.4 TCustomRibbonColorMap.CaptionFontColor Property

Pascal

```
property CaptionFontColor: TColor;
```

C++

```
__property TColor CaptionFontColor;
```

Description

This is CaptionFontColor, a member of class TCustomRibbonColorMap.

62.1.5.2.5 TCustomRibbonColorMap.ControlColor Property

Pascal

```
property ControlColor: TColor;
```

C++

```
__property TColor ControlColor;
```

Description

This is ControlColor, a member of class TCustomRibbonColorMap.

62.1.5.2.6 TCustomRibbonColorMap.ControlFrameColor Property

Pascal

```
property ControlFrameColor: TColor;
```

C++

```
__property TColor ControlFrameColor;
```

Description

This is ControlFrameColor, a member of class TCustomRibbonColorMap.

62.1.5.2.7 TCustomRibbonColorMap.ControlInactiveColor Property

Pascal

```
property ControlInactiveColor: TColor;
```

C++

```
__property TColor ControlInactiveColor;
```

Description

This is ControlInactiveColor, a member of class TCustomRibbonColorMap.

62.1.5.2.8 TCustomRibbonColorMap.DocumentFontColor Property

Pascal

```
property DocumentFontColor: TColor;
```

C++

```
__property TColor DocumentFontColor;
```

Description

This is DocumentFontColor, a member of class TCustomRibbonColorMap.

62.1.5.2.9 TCustomRibbonColorMap.GroupFontColor Property

Pascal

```
property GroupFontColor: TColor;
```

C++

```
__property TColor GroupFontColor;
```

Description

This is GroupFontColor, a member of class TCustomRibbonColorMap.

62.1.5.2.10 TCustomRibbonColorMap.InactiveTabFontColor Property

Pascal

```
property InactiveTabFontColor: TColor;
```

C++

```
__property TColor InactiveTabFontColor;
```

Description

This is InactiveTabFontColor, a member of class TCustomRibbonColorMap.

62.1.5.2.11 TCustomRibbonColorMap.QuickAccessToolbarMoreBorderColor Property

Pascal

```
property QuickAccessToolbarMoreBorderColor: TColor;
```

C++

```
__property TColor QuickAccessToolbarMoreBorderColor;
```

Description

This is QuickAccessToolbarMoreBorderColor, a member of class TCustomRibbonColorMap.

62.1.5.2.12 TCustomRibbonColorMap.QuickAccessToolbarMoreColor Property

Pascal

```
property QuickAccessToolbarMoreColor: TColor;
```

C++

```
__property TColor QuickAccessToolbarMoreColor;
```

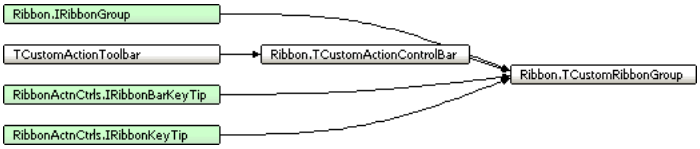
Description

This is QuickAccessToolbarMoreColor, a member of class TCustomRibbonColorMap.

62.1.6 TCustomRibbonGroup Class

Align and display all controls on ribbon.

Class Hierarchy



Pascal

```
TCustomRibbonGroup = class(TCustomActionControlBar, IRibbonGroup, IRibbonBarKeyTip,
IRibbonKeyTip);
```

C++

```
class TCustomRibbonGroup : public TCustomActionControlBar, public IRibbonGroup, public
IRibbonBarKeyTip, public IRibbonKeyTip;
```

File

Ribbon

Description

TCustomRibbonGroup aligns and displays all controls available for use on the ribbon. Controls are displayed on a Ribbon (see

page 3347) Group.

See Also

Creating an Application that Uses Ribbon Controls

Adding Commands to the Ribbon

TGroupAlign (see page 3408)

62.1.6.1 TCustomRibbonGroup Members

The following tables list the members exposed by TCustomRibbonGroup.

Enumerations

	Name	Description
	TGroupAlign (see page 3408)	Alignment of ribbon group.

Protected Methods

	Name	Description
	AlignControls (see page 3409)	This is AlignControls, a member of class TCustomRibbonGroup.
	AutoSizingChanged (see page 3410)	This is AutoSizingChanged, a member of class TCustomRibbonGroup.
	CMControlChange (see page 3357)	This is CMControlChange, a member of class TCustomActionControlBar.
	CMFontchanged (see page 3412)	This is CMFontchanged, a member of class TCustomRibbonGroup.
	CMItemDropPoint (see page 3412)	This is CMItemDropPoint, a member of class TCustomRibbonGroup.
	CMMouseenter (see page 3412)	This is CMMouseenter, a member of class TCustomRibbonGroup.
	CMMouseleave (see page 3412)	This is CMMouseleave, a member of class TCustomRibbonGroup.
	CMTextchanged (see page 3413)	This is CMTextchanged, a member of class TCustomRibbonGroup.
	CreateControls (see page 3413)	This is CreateControls, a member of class TCustomRibbonGroup.
	DoDropItem (see page 3414)	This is DoDropItem, a member of class TCustomRibbonGroup.
	DoEndDrag (see page 3414)	This is DoEndDrag, a member of class TCustomRibbonGroup.
	DrawShowDialogButton (see page 3414)	This is DrawShowDialogButton, a member of class TCustomRibbonGroup.
	GetCaptionHeight (see page 3414)	This is GetCaptionHeight, a member of class TCustomRibbonGroup.
	GetControlClass (see page 3357)	This is GetControlClass, a member of class TCustomActionControlBar.
	GetGroupHeight (see page 3415)	This is GetGroupHeight, a member of class TCustomRibbonGroup.
	HideControls (see page 3416)	This is HideControls, a member of class TCustomRibbonGroup.
	MouseDown (see page 3417)	This is MouseDown, a member of class TCustomRibbonGroup.
	MouseMove (see page 3417)	This is MouseMove, a member of class TCustomRibbonGroup.
	MouseUp (see page 3417)	This is MouseUp, a member of class TCustomRibbonGroup.
	Notification (see page 3417)	This is Notification, a member of class TCustomRibbonGroup.
	Paint (see page 3418)	This is Paint, a member of class TCustomRibbonGroup.
	Resize (see page 3418)	This is Resize, a member of class TCustomRibbonGroup.
	SetAutoSizing (see page 3418)	This is SetAutoSizing, a member of class TCustomRibbonGroup.
	SetParent (see page 3419)	This is SetParent, a member of class TCustomRibbonGroup.
	SetupDropDownBtn (see page 3419)	This is SetupDropDownBtn, a member of class TCustomRibbonGroup.
	ShowControls (see page 3419)	This is ShowControls, a member of class TCustomRibbonGroup.

Protected Properties

	Name	Description
	CreatingControls (see page 3358)	This is CreatingControls, a member of class TCustomActionControlBar.

Public Methods

	Name	Description
	AlignGroupControls (see page 3410)	Align controls in Ribbon (see page 3347) Group
	Assign (see page 3410)	This is Assign, a member of class TCustomRibbonGroup.

	BarHandleKeyTips (see page 3410)	Iterate over any visible **KeyTip (see page 3423)**s and perform validation to determine if one of **KeyTip (see page 3423)**s is partial or full match.
	BarHideKeyTips (see page 3411)	Hide any visible KeyTips that commands are displaying.
	BarShowKeyTips (see page 3411)	Show K**eyTip**s for commands.
	Clone (see page 3411)	This is Clone, a member of class TCustomRibbonGroup.
V	Create (see page 3407)	This is Create, a member of class TCustomRibbonGroup.
V	CreateControl (see page 3413)	Recreate specific control.
V	Destroy (see page 3408)	This is Destroy, a member of class TCustomRibbonGroup.
V	DockDrop (see page 3413)	This is DockDrop, a member of class TCustomRibbonGroup.
V	DragDrop (see page 3414)	This is DragDrop, a member of class TCustomRibbonGroup.
	GetKeyTip (see page 3415)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3415)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3416)	Hide **KeyTip** for command.
	KeyTipClick (see page 3416)	Handle clicking a **KeyTip**.
V	SetBounds (see page 3418)	This is SetBounds, a member of class TCustomRibbonGroup.
	SetKeyTip (see page 3418)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3419)	Shows **KeyTip** for command in specified position.
V	StyleChanged (see page 3420)	This is StyleChanged, a member of class TCustomRibbonGroup.

Public Nested Types

Name	Description
TRowPosition (see page 3424)	This is nested type Ribbon.TCustomRibbonGroup.TRowPosition.

Public Properties

	Name	Description
R	CanCollapse (see page 3421)	Determine whether group can be displayed collapsed.
R	CollapsedControl (see page 3421)	This is CollapsedControl, a member of class TCustomRibbonGroup.
	DialogAction (see page 3421)	Determines whether dialog button displayed in group.
	GroupAlign (see page 3421)	Determines how controls on group are aligned.
	GroupIndex (see page 3422)	Index of group within current tab.
	GroupVariant (see page 3422)	Current size variant ribbon group is using.
R	GroupWidth (see page 3422)	This is GroupWidth, a member of class TCustomRibbonGroup.
R	HasGallery (see page 3423)	Indicates if group has an in ribbon Gallery.
	Items (see page 3423)	This is Items, a member of class TCustomRibbonGroup.
	KeyTip (see page 3423)	Key tip for Dialog button of group.
R	Page (see page 3423)	Reference to Page group is on.
R	RequiredWidth (see page 3424)	Width required to display group.
	Rows (see page 3424)	Number of Rows in group has for controls.

62.1.6.2 TCustomRibbonGroup.Create Constructor

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomRibbonGroup(TComponent * AOwner);
```

Description

This is Create, a member of class TCustomRibbonGroup.

62.1.6.3 TCustomRibbonGroup.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomRibbonGroup();
```

Description

This is Destroy, a member of class TCustomRibbonGroup.

62.1.6.4 TCustomRibbonGroup Enumerations

The enumerations of the TCustomRibbonGroup class are listed here.

Enumerations

	Name	Description
	TGroupAlign (see page 3408)	Alignment of ribbon group.

62.1.6.4.1 Ribbon.TCustomRibbonGroup.TGroupAlign Enumeration

Alignment of ribbon group.

Pascal

```
TGroupAlign = (
  gaHorizontal,
  gaVertical
);
```

C++

```
enum TGroupAlign {
  gaHorizontal,
  gaVertical
};
```

File

Ribbon

Description

TGroupAlign specifies the alignment of the TCustomRibbonGroup (see page 3405). It can have one of the following values:

Value	Meaning
gaHorizontal	Controls are aligned left to right in rows. The row ends when a control has the NewRow property set to True. The NewRow property can be ignored when a group is being minimized.
gaVertical	Controls are aligned top to bottom moving left to right. Groups aligned vertically are autosized.

62.1.6.5 TCustomRibbonGroup Methods

The methods of the TCustomRibbonGroup class are listed here.

Protected Methods

	Name	Description
	AlignControls (see page 3409)	This is AlignControls, a member of class TCustomRibbonGroup.
	AutoSizingChanged (see page 3410)	This is AutoSizingChanged, a member of class TCustomRibbonGroup.
	CMFontchanged (see page 3412)	This is CMFontchanged, a member of class TCustomRibbonGroup.
	CMItemDropPoint (see page 3412)	This is CMItemDropPoint, a member of class TCustomRibbonGroup.
	CMMouseenter (see page 3412)	This is CMMouseenter, a member of class TCustomRibbonGroup.
	CMMouseleave (see page 3412)	This is CMMouseleave, a member of class TCustomRibbonGroup.
	CMTextchanged (see page 3413)	This is CMTextchanged, a member of class TCustomRibbonGroup.
	CreateControls (see page 3413)	This is CreateControls, a member of class TCustomRibbonGroup.
	DoDropItem (see page 3414)	This is DoDropItem, a member of class TCustomRibbonGroup.
	DoEndDrag (see page 3414)	This is DoEndDrag, a member of class TCustomRibbonGroup.
	DrawShowDialogButton (see page 3414)	This is DrawShowDialogButton, a member of class TCustomRibbonGroup.
	GetCaptionHeight (see page 3414)	This is GetCaptionHeight, a member of class TCustomRibbonGroup.
	GetGroupHeight (see page 3415)	This is GetGroupHeight, a member of class TCustomRibbonGroup.
	HideControls (see page 3416)	This is HideControls, a member of class TCustomRibbonGroup.
	MouseDown (see page 3417)	This is MouseDown, a member of class TCustomRibbonGroup.
	MouseMove (see page 3417)	This is MouseMove, a member of class TCustomRibbonGroup.
	MouseUp (see page 3417)	This is MouseUp, a member of class TCustomRibbonGroup.
	Notification (see page 3417)	This is Notification, a member of class TCustomRibbonGroup.
	Paint (see page 3418)	This is Paint, a member of class TCustomRibbonGroup.
	Resize (see page 3418)	This is Resize, a member of class TCustomRibbonGroup.
	SetAutoSizing (see page 3418)	This is SetAutoSizing, a member of class TCustomRibbonGroup.
	SetParent (see page 3419)	This is SetParent, a member of class TCustomRibbonGroup.
	SetupDropDownBtn (see page 3419)	This is SetupDropDownBtn, a member of class TCustomRibbonGroup.
	ShowControls (see page 3419)	This is ShowControls, a member of class TCustomRibbonGroup.

Public Methods

	Name	Description
	AlignGroupControls (see page 3410)	Align controls in Ribbon (see page 3347) Group
	Assign (see page 3410)	This is Assign, a member of class TCustomRibbonGroup.
	BarHandleKeyTips (see page 3410)	Iterate over any visible **KeyTip (see page 3423)**s and perform validation to determine if one of **KeyTip (see page 3423)**s is partial or full match.
	BarHideKeyTips (see page 3411)	Hide any visible KeyTips that commands are displaying.
	BarShowKeyTips (see page 3411)	Show K**eyTip**s for commands.
	Clone (see page 3411)	This is Clone, a member of class TCustomRibbonGroup.
	CreateControl (see page 3413)	Recreate specific control.
	DockDrop (see page 3413)	This is DockDrop, a member of class TCustomRibbonGroup.
	DragDrop (see page 3414)	This is DragDrop, a member of class TCustomRibbonGroup.
	GetKeyTip (see page 3415)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3415)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3416)	Hide **KeyTip** for command.
	KeyTipClick (see page 3416)	Handle clicking a **KeyTip**.
	SetBounds (see page 3418)	This is SetBounds, a member of class TCustomRibbonGroup.
	SetKeyTip (see page 3418)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3419)	Shows **KeyTip** for command in specified position.
	StyleChanged (see page 3420)	This is StyleChanged, a member of class TCustomRibbonGroup.

62.1.6.5.1 TCustomRibbonGroup.AlignControls Method

Pascal

```
procedure AlignControls(AControl: TControl; var Rect: TRect); override;
```

C++

```
virtual __fastcall AlignControls(TControl * AControl, TRect Rect);
```

Description

This is AlignControls, a member of class TCustomRibbonGroup.

62.1.6.5.2 TCustomRibbonGroup.AlignGroupControls Method

Align controls in Ribbon (see page 3347) Group

Pascal

```
procedure AlignGroupControls; virtual;
```

C++

```
virtual __fastcall AlignGroupControls();
```

Description

AlignGroupControls aligns the controls in the Ribbon (see page 3347) Group.

62.1.6.5.3 TCustomRibbonGroup.Assign Method

Pascal

```
procedure Assign(Source: TPersistent); override;
```

C++

```
virtual __fastcall Assign(TPersistent * Source);
```

Description

This is Assign, a member of class TCustomRibbonGroup.

62.1.6.5.4 TCustomRibbonGroup.AutoSizingChanged Method

Pascal

```
procedure AutoSizingChanged; override;
```

C++

```
virtual __fastcall AutoSizingChanged();
```

Description

This is AutoSizingChanged, a member of class TCustomRibbonGroup.

62.1.6.5.5 TCustomRibbonGroup.BarHandleKeyTips Method

Iterate over any visible **KeyTip (see page 3423)**s and perform validation to determine if one of **KeyTip (see page 3423)**s is partial or full match.

Pascal

```
procedure BarHandleKeyTips(const KeyTip: string; var KeyTips: TKeyTipList);
```

C++

```
__fastcall BarHandleKeyTips(const UnicodeString KeyTip, TKeyTipList KeyTips);
```

Description

BarHandleKeyTips iterates over any visible **KeyTip (see page 3423)**s and performs validation to determine if one of the **KeyTip (see page 3423)**s is a partial or full match. **KeyTip (see page 3423)**s that match are returned in the **List** parameter.

See Also

BarHideKeyTips (see page 3411)

BarShowKeyTips (see page 3411)

62.1.6.5.6 TCustomRibbonGroup.BarHideKeyTips Method

Hide any visible KeyTips that commands are displaying.

Pascal

```
procedure BarHideKeyTips(KeyTips: TKeyTipList);
```

C++

```
__fastcall BarHideKeyTips(TKeyTipList KeyTips);
```

Description

BarHideKeyTips hides any visible KeyTips that commands are displaying. The **KeyTips** parameter contains a list of TKeyTipMatchItem (see page 3598) objects that indicate the **KeyTip (see page 3423)** associate with that command should not be close. When the **KeyTips** parameter is `nil`, all **KeyTip (see page 3423)**s are hidden.

See Also

TKeyTipMatchItem (see page 3598)

BarHandleKeyTips (see page 3410)

BarShowKeyTips (see page 3411)

62.1.6.5.7 TCustomRibbonGroup.BarShowKeyTips Method

Show K**eyTip**s for commands.

Pascal

```
procedure BarShowKeyTips;
```

C++

```
__fastcall BarShowKeyTips();
```

Description

BarShowKeyTips shows **KeyTip (see page 3423)**s for the commands.

See Also

BarHandleKeyTips (see page 3410)

BarHideKeyTips (see page 3411)

62.1.6.5.8 TCustomRibbonGroup.Clone Method

Pascal

```
function Clone: TCustomRibbonGroup;
```

C++

```
__fastcall TCustomRibbonGroup Clone();
```

Description

This is Clone, a member of class TCustomRibbonGroup.

62.1.6.5.9 TCustomRibbonGroup.CMFontchanged Method

Pascal

```
procedure CMFontchanged(var Message: TMessage); message CM_FONTCHANGED;
```

C++

```
__fastcall CMFontchanged(TMessage Message);
```

Description

This is CMFontchanged, a member of class TCustomRibbonGroup.

62.1.6.5.10 TCustomRibbonGroup.CMItemDropPoint Method

Pascal

```
procedure CMItemDropPoint(var Message: TCMItemMsg); message CM_ITEMDROPPOINT;
```

C++

```
__fastcall CMItemDropPoint(TCMItemMsg Message);
```

Description

This is CMItemDropPoint, a member of class TCustomRibbonGroup.

62.1.6.5.11 TCustomRibbonGroup.CMMouseenter Method

Pascal

```
procedure CMMouseenter(var Message: TMessage); message CM_MOUSEENTER;
```

C++

```
__fastcall CMMouseenter(TMessage Message);
```

Description

This is CMMouseenter, a member of class TCustomRibbonGroup.

62.1.6.5.12 TCustomRibbonGroup.CMMouseleave Method

Pascal

```
procedure CMMouseleave(var Message: TMessage); message CM_MOUSELEAVE;
```

C++

```
__fastcall CMMouseleave(TMessage Message);
```

Description

This is CMMouseleave, a member of class TCustomRibbonGroup.

62.1.6.5.13 TCustomRibbonGroup.CMTextchanged Method

Pascal

```
procedure CMTextchanged(var Message: TMessage); message CM_TEXTCHANGED;
```

C++

```
__fastcall CMTextchanged(TMessage Message);
```

Description

This is CMTextchanged, a member of class TCustomRibbonGroup.

62.1.6.5.14 TCustomRibbonGroup.CreateControl Method

Recreate specific control.

Pascal

```
function CreateControl(AnItem: TActionClientItem): TCustomActionControl; override;
```

C++

```
virtual __fastcall TCustomActionControl CreateControl(TActionClientItem AnItem);
```

Description

CreateControl recreates a specific control.

See Also

AlignGroupControls (see page 3568)

62.1.6.5.15 TCustomRibbonGroup.CreateControls Method

Pascal

```
procedure CreateControls; override;
```

C++

```
virtual __fastcall CreateControls();
```

Description

This is CreateControls, a member of class TCustomRibbonGroup.

62.1.6.5.16 TCustomRibbonGroup.DockDrop Method

Pascal

```
procedure DockDrop(Source: TDragDockObject; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall DockDrop(TDragDockObject Source, int X, int Y);
```

Description

This is DockDrop, a member of class TCustomRibbonGroup.

62.1.6.5.17 TCustomRibbonGroup.DoDropItem Method

Pascal

```
procedure DoDropItem(Source: TActionItemDragObject; const X: Integer; const Y: Integer);
override;
```

C++

```
virtual __fastcall DoDropItem(TActionItemDragObject Source, const int X, const int Y);
```

Description

This is DoDropItem, a member of class TCustomRibbonGroup.

62.1.6.5.18 TCustomRibbonGroup.DoEndDrag Method

Pascal

```
procedure DoEndDrag(Target: TObject; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall DoEndDrag(TObject * Target, int X, int Y);
```

Description

This is DoEndDrag, a member of class TCustomRibbonGroup.

62.1.6.5.19 TCustomRibbonGroup.DragDrop Method

Pascal

```
procedure DragDrop(Source: TObject; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall DragDrop(TObject * Source, int X, int Y);
```

Description

This is DragDrop, a member of class TCustomRibbonGroup.

62.1.6.5.20 TCustomRibbonGroup.DrawShowDialogButton Method

Pascal

```
procedure DrawShowDialogButton; virtual;
```

C++

```
virtual __fastcall DrawShowDialogButton();
```

Description

This is DrawShowDialogButton, a member of class TCustomRibbonGroup.

62.1.6.5.21 TCustomRibbonGroup.GetCaptionHeight Method

Pascal

```
function GetCaptionHeight: Integer;
```

C++

```
__fastcall int GetCaptionHeight();
```

Description

This is GetCaptionHeight, a member of class TCustomRibbonGroup.

62.1.6.5.22 TCustomRibbonGroup.GetGroupHeight Method

Pascal

```
function GetGroupHeight: Integer;
```

C++

```
__fastcall int GetGroupHeight();
```

Description

This is GetGroupHeight, a member of class TCustomRibbonGroup.

62.1.6.5.23 TCustomRibbonGroup.GetKeyTip Method

Returns current **KeyTip** for command.

Pascal

```
function GetKeyTip: string;
```

C++

```
__fastcall UnicodeString GetKeyTip();
```

Description

GetKeyTip returns the current **KeyTip** for the command

See Also

KeyTip (see page 3922)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

62.1.6.5.24 TCustomRibbonGroup.HandleKeyTip Method

Indicates if command's **KeyTip** matches currently input KeyTip.

Pascal

```
procedure HandleKeyTip(const KeyTip: string; var Handled: TKeyTipHandled);
```

C++

```
__fastcall HandleKeyTip(const UnicodeString KeyTip, TKeyTipHandled Handled);
```

Description

HandleKeyTip indicates whether this command's

KeyTip matches the currently input KeyTip by the user. Full and partial matches are taken into consideration.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

62.1.6.5.25 TCustomRibbonGroup.HideControls Method

Pascal

```
procedure HideControls; virtual;
```

C++

```
virtual __fastcall HideControls();
```

Description

This is HideControls, a member of class TCustomRibbonGroup.

62.1.6.5.26 TCustomRibbonGroup.HideKeyTip Method

Hide **KeyTip** for command.

Pascal

```
procedure HideKeyTip(const Match: string);
```

C++

```
__fastcall HideKeyTip(const UnicodeString Match);
```

Description

This method hides the **KeyTip** for this command. If the **Match** parameter matches (full or partial) the command's **KeyTip**, the **KeyTip** remains visible.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

62.1.6.5.27 TCustomRibbonGroup.KeyTipClick Method

Handle clicking a **KeyTip**.

Pascal

```
procedure KeyTipClick;
```

C++

```
__fastcall KeyTipClick();
```

Description

KeyTipClick Handle clicking a **KeyTip**. Some requires need to be handled differently when clicked by a **KeyTip**.

62.1.6.5.28 TCustomRibbonGroup.MouseDown Method

Pascal

```
procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer);
override;
```

C++

```
virtual __fastcall MouseDown(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

This is MouseDown, a member of class TCustomRibbonGroup.

62.1.6.5.29 TCustomRibbonGroup.MouseMove Method

Pascal

```
procedure MouseMove(Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseMove(TShiftState Shift, int X, int Y);
```

Description

This is MouseMove, a member of class TCustomRibbonGroup.

62.1.6.5.30 TCustomRibbonGroup.MouseUp Method

Pascal

```
procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseUp(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

This is MouseUp, a member of class TCustomRibbonGroup.

62.1.6.5.31 TCustomRibbonGroup.Notification Method

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

This is Notification, a member of class TCustomRibbonGroup.

62.1.6.5.32 TCustomRibbonGroup.Paint Method

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

This is Paint, a member of class TCustomRibbonGroup.

62.1.6.5.33 TCustomRibbonGroup.Resize Method

Pascal

```
procedure Resize; override;
```

C++

```
virtual __fastcall Resize();
```

Description

This is Resize, a member of class TCustomRibbonGroup.

62.1.6.5.34 TCustomRibbonGroup.SetAutoSizing Method

Pascal

```
procedure SetAutoSizing(const Value: Boolean); override;
```

C++

```
virtual __fastcall SetAutoSizing(const Boolean Value);
```

Description

This is SetAutoSizing, a member of class TCustomRibbonGroup.

62.1.6.5.35 TCustomRibbonGroup.SetBounds Method

Pascal

```
procedure SetBounds(ALeft: Integer; ATop: Integer; AWidth: Integer; AHeight: Integer);
override;
```

C++

```
virtual __fastcall SetBounds(int ALeft, int ATop, int AWidth, int AHeight);
```

Description

This is SetBounds, a member of class TCustomRibbonGroup.

62.1.6.5.36 TCustomRibbonGroup.SetKeyTip Method

Sets **KeyTip** for command.

Pascal

```
procedure SetKeyTip(const Value: string);
```

C++

```
__fastcall SetKeyTip(const UnicodeString Value);
```

Description

This method sets the **KeyTip** for the command.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

ShowKeyTip (see page 3921)

62.1.6.5.37 TCustomRibbonGroup.SetParent Method

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

This is SetParent, a member of class TCustomRibbonGroup.

62.1.6.5.38 TCustomRibbonGroup.SetupDropDownBtn Method

Pascal

```
procedure SetupDropDownBtn; override;
```

C++

```
virtual __fastcall SetupDropDownBtn();
```

Description

This is SetupDropDownBtn, a member of class TCustomRibbonGroup.

62.1.6.5.39 TCustomRibbonGroup.ShowControls Method

Pascal

```
procedure ShowControls; virtual;
```

C++

```
virtual __fastcall ShowControls();
```

Description

This is ShowControls, a member of class TCustomRibbonGroup.

62.1.6.5.40 TCustomRibbonGroup.ShowKeyTip Method

Shows **KeyTip** for command in specified position.

Pascal

```
procedure ShowKeyTip(const Position: TKeyTipPosition);
```

C++

```
__fastcall ShowKeyTip(const TKeyTipPosition Position);
```

Description

This method shows the **KeyTip** for the command in the position specified. Only the Application Button uses a **KeyTip** position.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

62.1.6.5.41 TCustomRibbonGroup.StyleChanged Method

Pascal

```
procedure StyleChanged(AStyle: TRibbonStyleActionBars); virtual;
```

C++

```
virtual __fastcall StyleChanged(TRibbonStyleActionBars AStyle);
```

Description

This is StyleChanged, a member of class TCustomRibbonGroup.

62.1.6.6 TCustomRibbonGroup Properties

The properties of the TCustomRibbonGroup class are listed here.

Public Properties

	Name	Description
R	CanCollapse (see page 3421)	Determine whether group can be displayed collapsed.
R	CollapsedControl (see page 3421)	This is CollapsedControl, a member of class TCustomRibbonGroup.
	DialogAction (see page 3421)	Determines whether dialog button displayed in group.
	GroupAlign (see page 3421)	Determines how controls on group are aligned.
	GroupIndex (see page 3422)	Index of group within current tab.
	GroupVariant (see page 3422)	Current size variant ribbon group is using.
R	GroupWidth (see page 3422)	This is GroupWidth, a member of class TCustomRibbonGroup.
R	HasGallery (see page 3423)	Indicates if group has an in ribbon Gallery.
	Items (see page 3423)	This is Items, a member of class TCustomRibbonGroup.
	KeyTip (see page 3423)	Key tip for Dialog button of group.
R	Page (see page 3423)	Reference to Page group is on.
R	RequiredWidth (see page 3424)	Width required to display group.
	Rows (see page 3424)	Number of Rows in group has for controls.

62.1.6.6.1 TCustomRibbonGroup.CanCollapse Property

Determine whether group can be displayed collapsed.

Pascal

```
property CanCollapse: Boolean;
```

C++

```
__property Boolean CanCollapse;
```

Description

CanCollapse determines whether the group can be displayed in the collapsed state. If there is only one control on the group, that group cannot be displayed as collapsed.

62.1.6.6.2 TCustomRibbonGroup.CollapsedControl Property

Pascal

```
property CollapsedControl: TRibbonGroupCollapsedControl;
```

C++

```
__property TRibbonGroupCollapsedControl CollapsedControl;
```

Description

This is CollapsedControl, a member of class TCustomRibbonGroup.

62.1.6.6.3 TCustomRibbonGroup.DialogAction Property

Determines whether dialog button displayed in group.

Pascal

```
property DialogAction: TContainedAction;
```

C++

```
__property TContainedAction DialogAction;
```

Description

DialogAction determines whether the dialog button is displayed in the lower right hand corner of the group.

See Also

KeyTip (see page 3423)

62.1.6.6.4 TCustomRibbonGroup.GroupAlign Property

Determines how controls on group are aligned.

Pascal

```
property GroupAlign: TGroupAlign;
```

C++

```
__property TGroupAlign GroupAlign;
```

Description

GroupAlign contains a TGroupAlign (see page 3408) that determines how controls on a group are aligned.

If alignment is **gaHorizontal**, controls are aligned left to right, top to bottom. To make a control start in a new row, set the NewRow property to true.

If alignment is **gaVertical**, controls are aligned top to bottom, left to right. Controls are automatically moved to the next column in the group if three controls are already in the column. You can also force a control to a new column by setting the NewCol property to true.

See Also

TGroupAlign (see page 3408)

62.1.6.6.5 TCustomRibbonGroup.GroupIndex Property

Index of group within current tab.

Pascal

```
property GroupIndex: Integer;
```

C++

```
__property int GroupIndex;
```

Description

GroupIndex is the index of the group within the current tab. Set the GroupIndex property to change the position of the group within the current tab.

62.1.6.6.6 TCustomRibbonGroup.GroupVariant Property

Current size variant ribbon group is using.

Pascal

```
property GroupVariant: TRibbonGroupVariant;
```

C++

```
__property TRibbonGroupVariant GroupVariant;
```

Description

GroupVariant is the current size variant the ribbon group is using.

See Also

RequiredWidth (see page 3424)

62.1.6.6.7 TCustomRibbonGroup.GroupWidth Property

Pascal

```
property GroupWidth: Integer;
```

C++

```
__property int GroupWidth;
```

Description

This is GroupWidth, a member of class TCustomRibbonGroup.

62.1.6.6.8 TCustomRibbonGroup.HasGallery Property

Indicates if group has an in ribbon Gallery.

Pascal

```
property HasGallery: Boolean;
```

C++

```
__property Boolean HasGallery;
```

Description

HasGallery returns True if the group has an in ribbon Gallery to display, false otherwise.

62.1.6.6.9 TCustomRibbonGroup.Items Property

Pascal

```
property Items;
```

C++

```
__property Items;
```

Description

This is Items, a member of class TCustomRibbonGroup.

62.1.6.6.10 TCustomRibbonGroup.KeyTip Property

Key tip for Dialog button of group.

Pascal

```
property KeyTip: string;
```

C++

```
__property UnicodeString KeyTip;
```

Description

KeyTip is the key tip for the Dialog button of the group. It is only valid when DialogAction (see page 3421) has been set.

See Also

DialogAction (see page 3421)

62.1.6.6.11 TCustomRibbonGroup.Page Property

Reference to Page group is on.

Pascal

```
property Page: TRibbonPage;
```

C++

```
__property TRibbonPage Page;
```

Description

Page is a reference to the Page that the group is on.

62.1.6.6.12 TCustomRibbonGroup.RequiredWidth Property

Width required to display group.

Pascal

```
property RequiredWidth: Integer;
```

C++

```
__property int RequiredWidth;
```

Description

RequiredWidth is the width required to display the group in its large Variant.

See Also

GroupVariant (see page 3422)

62.1.6.6.13 TCustomRibbonGroup.Rows Property

Number of Rows in group has for controls.

Pascal

```
property Rows: Integer;
```

C++

```
__property int Rows;
```

Description

Rows is the number of Rows the group has for displaying controls. A group must have at least 1 row and a maximum of 3.

62.1.6.7 TCustomRibbonGroup Nested Types

The nested types of the TCustomRibbonGroup class are listed here.

Public Nested Types

Name	Description
TRowPosition (see page 3424)	This is nested type Ribbon.TCustomRibbonGroup.TRowPosition.

62.1.6.7.1 TCustomRibbonGroup.TRowPosition Nested Type

Pascal

```
TRowPosition = array [1..3] of Integer;
```

C++

```
array [1..3] of Integer TRowPosition;
```

Description

This is nested type Ribbon.TCustomRibbonGroup.TRowPosition.

62.1.7 TCustomRibbonPage Class

Holds groups of controls.

Class Hierarchy

Pascal

```
TCustomRibbonPage = class(TCustomControl);
```

C++

```
class TCustomRibbonPage : public TCustomControl;
```

File

Ribbon

Description

TCustomRibbonPage holds the groups of controls that are displayed.

See Also

Creating an Application that Uses Ribbon Controls

Adding Commands to the Ribbon

62.1.7.1 TCustomRibbonPage Members

The following tables list the members exposed by TCustomRibbonPage.

Protected Methods

	Name	Description
	CMControlListChanging (see page 3429)	This is CMControlListChanging, a member of class TCustomRibbonPage.
	CMTextchanged (see page 3429)	This is CMTextchanged, a member of class TCustomRibbonPage.
V	CreateScrollButton (see page 3430)	This is CreateScrollButton, a member of class TCustomRibbonPage.
V	DoScrollButtonClick (see page 3430)	This is DoScrollButtonClick, a member of class TCustomRibbonPage.
	GetPageHeight (see page 3431)	This is GetPageHeight, a member of class TCustomRibbonPage.
	GetPageTop (see page 3431)	This is GetPageTop, a member of class TCustomRibbonPage.
V	Loaded (see page 3432)	Initializes the control after it is loaded from a stream.
V	Notification (see page 3433)	Responds to notifications that components are being created or destroyed.
V	Paint (see page 3433)	Renders the image of a custom control.
V	RequestAlign (see page 3434)	Instructs the parent of a control to reposition the control, enforcing its Align property.
V	Resize (see page 3434)	Respond to control resize.
V	SetParent (see page 3435)	Sets the parent of the control.
	UpdateScrollButtons (see page 3436)	This is UpdateScrollButtons, a member of class TCustomRibbonPage.

Protected Properties

	Name	Description
	GroupOffset (see page 3437)	This is GroupOffset, a member of class TCustomRibbonPage.

Public Constants

	Name	Description
	cRibbonGroupLeftOffset (see page 3428)	This is cRibbonGroupLeftOffset, a member of class TCustomRibbonPage.
	cRibbonGroupSpacing (see page 3428)	This is cRibbonGroupSpacing, a member of class TCustomRibbonPage.

Public Methods

	Name	Description
	AddGroup (see page 3429)	This is AddGroup, a member of class TCustomRibbonPage.
	AlignGroups (see page 3429)	This is AlignGroups, a member of class TCustomRibbonPage.
	CollapseGroup (see page 3430)	This is CollapseGroup, a member of class TCustomRibbonPage.
V	Create (see page 3426)	Creates an instance of TCustomControl.
V	Destroy (see page 3427)	Destroys an instance of TCustomControl.
	FindGroup (see page 3430)	This is FindGroup, a member of class TCustomRibbonPage.
	FindGroupAt (see page 3430)	This is FindGroupAt, a member of class TCustomRibbonPage.
	GetGroupPos (see page 3431)	This is GetGroupPos, a member of class TCustomRibbonPage.
	GetNextVisibleGroup (see page 3431)	This is GetNextVisibleGroup, a member of class TCustomRibbonPage.
	HandleKeyTips (see page 3432)	This is HandleKeyTips, a member of class TCustomRibbonPage.
	HideKeyTips (see page 3432)	This is HideKeyTips, a member of class TCustomRibbonPage.
V	Invalidate (see page 3432)	Schedules a control repaint.
	RibbonStyle (see page 3434)	Return style class to use when drawing specific Ribbon (see page 3438) elements.
V	SetBounds (see page 3435)	Sets the windowed control's boundary properties all at once.
	ShowKeyTips (see page 3436)	This is ShowKeyTips, a member of class TCustomRibbonPage.
	UpdateActionManager (see page 3436)	This is UpdateActionManager, a member of class TCustomRibbonPage.

Public Properties

	Name	Description
R	CollapseGroupCount (see page 3437)	This is CollapseGroupCount, a member of class TCustomRibbonPage.
R	GroupCount (see page 3437)	This is GroupCount, a member of class TCustomRibbonPage.
R	Groups (see page 3437)	This is Groups, a member of class TCustomRibbonPage.
	Index (see page 3438)	This is Index, a member of class TCustomRibbonPage.
	KeyTip (see page 3438)	This is KeyTip, a member of class TCustomRibbonPage.
	Ribbon (see page 3438)	This is Ribbon, a member of class TCustomRibbonPage.

62.1.7.2 TCustomRibbonPage.Create Constructor

Creates an instance of TCustomControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomRibbonPage(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomControl. However, you should never attempt to instantiate a TCustomControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants. Controls placed on forms at design time are created automatically.

After calling the inherited constructor, Create initializes the control, including the following property settings:

Initializes a TCanvas object for its Canvas property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Create

Destroy

Canvas

Owner

62.1.7.3 TCustomRibbonPage.Destroy Destructor

Destroys an instance of TCustomControl.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomRibbonPage();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the control is not nil, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

As the control is destroyed, it:

Destroys the TCanvas object in its Canvas property.

Override Destroy to free any memory or resources allocated in the Create method. When declaring a Destroy method in a descendant control type, always add the override directive to the declaration and call the inherited Destroy as the last statement in the redeclared method.

When a control is created, Delphi automatically provides exception handling around the constructor call. The destructor is called if an exception escapes from the constructor. This means that the destructor code must be prepared to clean up a partially constructed instance. Check all data fields for zero before disposing of their contents.

See Also

System::TObject::Free

delete

Create

62.1.7.4 TCustomRibbonPage Constants

The constants of the TCustomRibbonPage class are listed here.

Public Constants

	Name	Description
	cRibbonGroupLeftOffset (see page 3428)	This is cRibbonGroupLeftOffset, a member of class TCustomRibbonPage.
	cRibbonGroupSpacing (see page 3428)	This is cRibbonGroupSpacing, a member of class TCustomRibbonPage.

62.1.7.4.1 TCustomRibbonPage.cRibbonGroupLeftOffset Constant

Pascal

```
const cRibbonGroupLeftOffset = 4;
```

C++

```
cRibbonGroupLeftOffset = 4;
```

Description

This is cRibbonGroupLeftOffset, a member of class TCustomRibbonPage.

62.1.7.4.2 TCustomRibbonPage.cRibbonGroupSpacing Constant

Pascal

```
const cRibbonGroupSpacing = 2;
```

C++

```
cRibbonGroupSpacing = 2;
```

Description

This is cRibbonGroupSpacing, a member of class TCustomRibbonPage.

62.1.7.5 TCustomRibbonPage Methods

The methods of the TCustomRibbonPage class are listed here.

Protected Methods

	Name	Description
	CMControlListChanging (see page 3429)	This is CMControlListChanging, a member of class TCustomRibbonPage.
	CMTextchanged (see page 3429)	This is CMTextchanged, a member of class TCustomRibbonPage.
V	CreateScrollButton (see page 3430)	This is CreateScrollButton, a member of class TCustomRibbonPage.
V	DoScrollButtonClick (see page 3430)	This is DoScrollButtonClick, a member of class TCustomRibbonPage.
	GetPageHeight (see page 3431)	This is GetPageHeight, a member of class TCustomRibbonPage.
	GetPageTop (see page 3431)	This is GetPageTop, a member of class TCustomRibbonPage.
V	Loaded (see page 3432)	Initializes the control after it is loaded from a stream.
V	Notification (see page 3433)	Responds to notifications that components are being created or destroyed.
V	Paint (see page 3433)	Renders the image of a custom control.
V	RequestAlign (see page 3434)	Instructs the parent of a control to reposition the control, enforcing its Align property.
V	Resize (see page 3434)	Respond to control resize.
V	SetParent (see page 3435)	Sets the parent of the control.
	UpdateScrollButtons (see page 3436)	This is UpdateScrollButtons, a member of class TCustomRibbonPage.

Public Methods

	Name	Description
	AddGroup (see page 3429)	This is AddGroup, a member of class TCustomRibbonPage.
	AlignGroups (see page 3429)	This is AlignGroups, a member of class TCustomRibbonPage.
	CollapseGroup (see page 3430)	This is CollapseGroup, a member of class TCustomRibbonPage.
	FindGroup (see page 3430)	This is FindGroup, a member of class TCustomRibbonPage.
	FindGroupAt (see page 3430)	This is FindGroupAt, a member of class TCustomRibbonPage.
	GetGroupPos (see page 3431)	This is GetGroupPos, a member of class TCustomRibbonPage.
	GetNextVisibleGroup (see page 3431)	This is GetNextVisibleGroup, a member of class TCustomRibbonPage.

	HandleKeyTips (see page 3432)	This is HandleKeyTips, a member of class TCustomRibbonPage.
	HideKeyTips (see page 3432)	This is HideKeyTips, a member of class TCustomRibbonPage.
	Invalidate (see page 3432)	Schedules a control repaint.
	RibbonStyle (see page 3434)	Return style class to use when drawing specific Ribbon (see page 3438) elements.
	SetBounds (see page 3435)	Sets the windowed control's boundary properties all at once.
	ShowKeyTips (see page 3436)	This is ShowKeyTips, a member of class TCustomRibbonPage.
	UpdateActionManager (see page 3436)	This is UpdateActionManager, a member of class TCustomRibbonPage.

62.1.7.5.1 TCustomRibbonPage.AddGroup Method

Pascal

```
procedure AddGroup(const Group: TCustomRibbonGroup);
```

C++

```
__fastcall AddGroup(const TCustomRibbonGroup Group);
```

Description

This is AddGroup, a member of class TCustomRibbonPage.

62.1.7.5.2 TCustomRibbonPage.AlignGroups Method

Pascal

```
procedure AlignGroups;
```

C++

```
__fastcall AlignGroups();
```

Description

This is AlignGroups, a member of class TCustomRibbonPage.

62.1.7.5.3 TCustomRibbonPage.CMControlListChanging Method

Pascal

```
procedure CMControlListChanging(var Message: TCMControlListChanging); message
CM_CONTROLLISTCHANGING;
```

C++

```
__fastcall CMControlListChanging(TCMControlListChanging Message);
```

Description

This is CMControlListChanging, a member of class TCustomRibbonPage.

62.1.7.5.4 TCustomRibbonPage.CMTextchanged Method

Pascal

```
procedure CMTextchanged(var Message: TMessage); message CM_TEXTCHANGED;
```

C++

```
__fastcall CMTextchanged(TMessage Message);
```

Description

This is CMTextchanged, a member of class TCustomRibbonPage.

62.1.7.5.5 TCustomRibbonPage.CollapseGroup Method

Pascal

```
procedure CollapseGroup(const Group: TCustomRibbonGroup);
```

C++

```
__fastcall CollapseGroup(const TCustomRibbonGroup Group);
```

Description

This is CollapseGroup, a member of class TCustomRibbonPage.

62.1.7.5.6 TCustomRibbonPage.CreateScrollButton Method

Pascal

```
function CreateScrollButton(const Direction: TScrollDirection): TRibbonGroupScrollButton;
virtual;
```

C++

```
virtual __fastcall TRibbonGroupScrollButton CreateScrollButton(const TScrollDirection
Direction);
```

Description

This is CreateScrollButton, a member of class TCustomRibbonPage.

62.1.7.5.7 TCustomRibbonPage.DoScrollButtonClick Method

Pascal

```
procedure DoScrollButtonClick(const Direction: TScrollDirection); virtual;
```

C++

```
virtual __fastcall DoScrollButtonClick(const TScrollDirection Direction);
```

Description

This is DoScrollButtonClick, a member of class TCustomRibbonPage.

62.1.7.5.8 TCustomRibbonPage.FindGroup Method

Pascal

```
function FindGroup(const GroupCaption: string): TControl;
```

C++

```
__fastcall TControl * FindGroup(const UnicodeString GroupCaption);
```

Description

This is FindGroup, a member of class TCustomRibbonPage.

62.1.7.5.9 TCustomRibbonPage.FindGroupAt Method

Pascal

```
function FindGroupAt(const Pt: TPoint): TCustomRibbonGroup;
```

C++

```
__fastcall TCustomRibbonGroup FindGroupAt(const TPoint Pt);
```

Description

This is FindGroupAt, a member of class TCustomRibbonPage.

62.1.7.5.10 TCustomRibbonPage.GetGroupPos Method

Pascal

```
function GetGroupPos(Ignore: TControl): Integer;
```

C++

```
__fastcall int GetGroupPos(TControl * Ignore);
```

Description

This is GetGroupPos, a member of class TCustomRibbonPage.

62.1.7.5.11 TCustomRibbonPage.GetNextVisibleGroup Method

Pascal

```
function GetNextVisibleGroup: TCustomRibbonGroup;
```

C++

```
__fastcall TCustomRibbonGroup GetNextVisibleGroup();
```

Description

This is GetNextVisibleGroup, a member of class TCustomRibbonPage.

62.1.7.5.12 TCustomRibbonPage.GetPageHeight Method

Pascal

```
function GetPageHeight: Integer;
```

C++

```
__fastcall int GetPageHeight();
```

Description

This is GetPageHeight, a member of class TCustomRibbonPage.

62.1.7.5.13 TCustomRibbonPage.GetPageTop Method

Pascal

```
function GetPageTop: Integer;
```

C++

```
__fastcall int GetPageTop();
```

Description

This is GetPageTop, a member of class TCustomRibbonPage.

62.1.7.5.14 TCustomRibbonPage.HandleKeyTips Method

Pascal

```
procedure HandleKeyTips(const KeyTip: string; var KeyTips: TKeyTipList);
```

C++

```
__fastcall HandleKeyTips(const UnicodeString KeyTip, TKeyTipList KeyTips);
```

Description

This is HandleKeyTips, a member of class TCustomRibbonPage.

62.1.7.5.15 TCustomRibbonPage.HideKeyTips Method

Pascal

```
procedure HideKeyTips(KeyTips: TKeyTipList);
```

C++

```
__fastcall HideKeyTips(TKeyTipList KeyTips);
```

Description

This is HideKeyTips, a member of class TCustomRibbonPage.

62.1.7.5.16 TCustomRibbonPage.Invalidate Method

Schedules a control repaint.

Pascal

```
procedure Invalidate; override;
```

C++

```
virtual __fastcall Invalidate();
```

Description

Invalidate informs a control that its entire surface needs to be repainted. Calling Invalidate can prevent flicker caused by a series of partial repaints. There is no performance penalty for calling Invalidate multiple times before the control is actually repainted.

The actual repaint does not occur until the control is updated. To force an immediate repaint, call Repaint instead.

See Also

Repaint

Update

62.1.7.5.17 TCustomRibbonPage.Loaded Method

Initializes the control after it is loaded from a stream.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

Do not call Loaded. The VCL streaming system calls Loaded automatically after the control's form is loaded into memory so that the control can complete any initializations that depend on other objects in the form.

Loaded overrides the inherited method in order to initialize the control from its associated Action. To change the properties the control copies from its action, override the ActionChange method.

See Also

Loaded

Action

ActionChange

62.1.7.5.18 TCustomRibbonPage.Notification Method

Responds to notifications that components are being created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification allows TControl to update its PopupMenu property when the pop-up menu it refers to is destroyed.

Override the Notification method to respond to notifications that other components are about to be destroyed or have just been created. Use the Notification method to update controls that rely on other objects. Remove references to objects that are going away so that the control does not refer to objects that have been destroyed. For example, data-aware objects override the Notification method to update themselves when their data source is removed. Some objects respond to notifications that other objects of a specific type are being created. For example, the session component sets the Session property of new data-aware controls to itself when the AutoSessionName property is true.

By default, components pass along the notification to their owned components, if any.

See Also

PopupMenu

Notification

62.1.7.5.19 TCustomRibbonPage.Paint Method

Renders the image of a custom control.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method for TCustomControl does nothing other than provide the interface for a method that responds to WM_PAINT messages. Paint is called by the PaintWindow method, after that method has supplied the Canvas with the handle to a device context. When creating a custom control, always override Paint to draw the image of the control.

Tip: To determine which portions of the control's canvas need to be repainted when Paint is called, use the ClipRect property of the canvas.

See Also

Canvas

PaintWindow

GetDeviceContext

ClipRect

62.1.7.5.20 TCustomRibbonPage.RequestAlign Method

Instructs the parent of a control to reposition the control, enforcing its Align property.

Pascal

```
procedure RequestAlign; override;
```

C++

```
virtual __fastcall RequestAlign();
```

Description

Controls call RequestAlign internally when changes are made to the size or position of the control, so that they remain properly aligned within the parent control. Do not call RequestAlign in application code.

See Also

Align

Parent

62.1.7.5.21 TCustomRibbonPage.Resize Method

Respond to control resize.

Pascal

```
procedure Resize; override;
```

C++

```
virtual __fastcall Resize();
```

Description

Resize is called automatically immediately after the control's dimensions change. As implemented in TControl, Resize calls the control's OnResize event handler if it is assigned.

See Also

OnResize

62.1.7.5.22 TCustomRibbonPage.RibbonStyle Method

Return style class to use when drawing specific Ribbon (see page 3438) elements.

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

RibbonStyle returns the style class to use when drawing the specific Ribbon (see page 3438) elements.

62.1.7.5.23 TCustomRibbonPage.SetBounds Method

Sets the windowed control's boundary properties all at once.

Pascal

```
procedure SetBounds(ALeft: Integer; ATop: Integer; AWidth: Integer; AHeight: Integer);
override;
```

C++

```
virtual __fastcall SetBounds(int ALeft, int ATop, int AWidth, int AHeight);
```

Description

Use SetBounds to change all of the control's boundary properties at once. The same effect can be achieved by setting the Left, Top, Width, and Height properties separately. By setting all four properties at once, SetBounds ensures that the control will not repaint between changes

Specify the values for the Left, Top, Width, and Height properties as the value of the ALeft, ATop, AWidth, and AHeight parameters, respectively.

See Also

Height

Left

Top

Width

62.1.7.5.24 TCustomRibbonPage.SetParent Method

Sets the parent of the control.

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

SetParent is the protected implementation of the Parent property. Override SetParent to execute additional code when the value of the Parent property changes.

The AParent parameter specifies the new parent of the control.

If the control already has a parent, SetParent removes the control from that parent's list of controls by calling the parent's RemoveControl method. If AParent is not nil (Delphi) or NULL (C++), SetParent calls its InsertControl method to add the control to its list of controls.

The TControl destructor calls SetParent(nil (Delphi) or NULL (C++)) to remove the control from its parent's control list before destroying the component.

See Also

Destroy

InsertControl

Parent

RemoveControl

62.1.7.5.25 TCustomRibbonPage.ShowKeyTips Method

Pascal

```
procedure ShowKeyTips;
```

C++

```
__fastcall ShowKeyTips();
```

Description

This is ShowKeyTips, a member of class TCustomRibbonPage.

62.1.7.5.26 TCustomRibbonPage.UpdateActionManager Method

Pascal

```
procedure UpdateActionManager(Value: TCustomActionManager);
```

C++

```
__fastcall UpdateActionManager(TCustomActionManager Value);
```

Description

This is UpdateActionManager, a member of class TCustomRibbonPage.

62.1.7.5.27 TCustomRibbonPage.UpdateScrollButtons Method

Pascal

```
procedure UpdateScrollButtons;
```

C++

```
__fastcall UpdateScrollButtons();
```

Description

This is UpdateScrollButtons, a member of class TCustomRibbonPage.

62.1.7.6 TCustomRibbonPage Properties

The properties of the TCustomRibbonPage class are listed here.

Protected Properties

	Name	Description
	GroupOffset (see page 3437)	This is GroupOffset, a member of class TCustomRibbonPage.

Public Properties

	Name	Description
	CollapseGroupCount (see page 3437)	This is CollapseGroupCount, a member of class TCustomRibbonPage.
	GroupCount (see page 3437)	This is GroupCount, a member of class TCustomRibbonPage.
	Groups (see page 3437)	This is Groups, a member of class TCustomRibbonPage.
	Index (see page 3438)	This is Index, a member of class TCustomRibbonPage.
	KeyTip (see page 3438)	This is KeyTip, a member of class TCustomRibbonPage.
	Ribbon (see page 3438)	This is Ribbon, a member of class TCustomRibbonPage.

62.1.7.6.1 TCustomRibbonPage.CollapseGroupCount Property

Pascal

```
property CollapseGroupCount: Integer;
```

C++

```
__property int CollapseGroupCount;
```

Description

This is CollapseGroupCount, a member of class TCustomRibbonPage.

62.1.7.6.2 TCustomRibbonPage.GroupCount Property

Pascal

```
property GroupCount: Integer;
```

C++

```
__property int GroupCount;
```

Description

This is GroupCount, a member of class TCustomRibbonPage.

62.1.7.6.3 TCustomRibbonPage.GroupOffset Property

Pascal

```
property GroupOffset: Integer;
```

C++

```
__property int GroupOffset;
```

Description

This is GroupOffset, a member of class TCustomRibbonPage.

62.1.7.6.4 TCustomRibbonPage.Groups Property

Pascal

```
property Groups [Index: Integer]: TCustomRibbonGroup;
```

C++

```
__property TCustomRibbonGroup Groups[int Index];
```

Description

This is Groups, a member of class TCustomRibbonPage.

62.1.7.6.5 TCustomRibbonPage.Index Property

Pascal

```
property Index: Integer;
```

C++

```
__property int Index;
```

Description

This is Index, a member of class TCustomRibbonPage.

62.1.7.6.6 TCustomRibbonPage.KeyTip Property

Pascal

```
property KeyTip: string;
```

C++

```
__property UnicodeString KeyTip;
```

Description

This is KeyTip, a member of class TCustomRibbonPage.

62.1.7.6.7 TCustomRibbonPage.Ribbon Property

Pascal

```
property Ribbon: TCustomRibbon;
```

C++

```
__property TCustomRibbon Ribbon;
```

Description

This is Ribbon, a member of class TCustomRibbonPage.

62.1.8 TCustomRibbonQuickAccessPopupToolbar Class

Class Hierarchy

Pascal

```
TCustomRibbonQuickAccessPopupToolbar = class(TCustomActionPopupMenu);
```

C++

```
class TCustomRibbonQuickAccessPopupToolbar : public TCustomActionPopupMenu;
```

File

Ribbon

Description

This is class Ribbon.TCustomRibbonQuickAccessPopupToolbar.

62.1.8.1 TCustomRibbonQuickAccessPopupToolbar Members

The following tables list the members exposed by TCustomRibbonQuickAccessPopupToolbar.

Protected Methods

	Name	Description
	CreateWnd (see page 3440)	Creates the underlying window.
	DrawBackground (see page 3440)	This is DrawBackground, a member of class TCustomRibbonQuickAccessPopupToolbar.
	GetControlClass (see page 3441)	This is GetControlClass, a member of class TCustomRibbonQuickAccessPopupToolbar.
	NCPaint (see page 3441)	This is NCPaint, a member of class TCustomRibbonQuickAccessPopupToolbar.
	ValidOrientations (see page 3441)	This is ValidOrientations, a member of class TCustomRibbonQuickAccessPopupToolbar.

Public Methods

	Name	Description
	Create (see page 3439)	Creates and initializes a TCustomActionPopupMenu instance.
	CreateControl (see page 3440)	This is CreateControl, a member of class TCustomRibbonQuickAccessPopupToolbar.

62.1.8.2 TCustomRibbonQuickAccessPopupToolbar.Create Constructor

Creates and initializes a TCustomActionPopupMenu instance.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomRibbonQuickAccessPopupToolbar(TComponent * AOwner);
```

Description

Create generates a TCustomActionPopupMenu and initializes its properties.

```
__fastcall virtual TCustomActionMenuBar (Classes::TComponentTComponent_Object* AOwner);
__fastcall TCustomActionMenuBar (HWND ParentWindow) :
Controls::TWinControlTWinControl_Create(ParentWindow) { }
```

62.1.8.3 TCustomRibbonQuickAccessPopupToolbar Methods

The methods of the TCustomRibbonQuickAccessPopupToolbar class are listed here.

Protected Methods

	Name	Description
	CreateWnd (see page 3440)	Creates the underlying window.
	DrawBackground (see page 3440)	This is DrawBackground, a member of class TCustomRibbonQuickAccessPopupToolbar.
	GetControlClass (see page 3441)	This is GetControlClass, a member of class TCustomRibbonQuickAccessPopupToolbar.

	NCPaint (see page 3441)	This is NCPaint, a member of class TCustomRibbonQuickAccessPopupToolbar.
	ValidOrientations (see page 3441)	This is ValidOrientations, a member of class TCustomRibbonQuickAccessPopupToolbar.

Public Methods

	Name	Description
	CreateControl (see page 3440)	This is CreateControl, a member of class TCustomRibbonQuickAccessPopupToolbar.

62.1.8.3.1 TCustomRibbonQuickAccessPopupToolbar.CreateControl Method

Pascal

```
function CreateControl(AnItem: TActionClientItem): TCustomActionControl; override;
```

C++

```
virtual __fastcall TCustomActionControl CreateControl(TActionClientItem AnItem);
```

Description

This is CreateControl, a member of class TCustomRibbonQuickAccessPopupToolbar.

62.1.8.3.2 TCustomRibbonQuickAccessPopupToolbar.CreateWnd Method

Creates the underlying window.

Pascal

```
procedure CreateWnd; override;
```

C++

```
virtual __fastcall CreateWnd();
```

Description

CreateWnd is called automatically when the control is first created or when the underlying screen object must be destroyed and recreated to reflect property changes. Override CreateWnd to pass additional initialization messages to the screen object when it is created.

See Also

CreateParams

CreateWindowHandle

Perform

DestroyWnd

62.1.8.3.3 TCustomRibbonQuickAccessPopupToolbar.DrawBackground Method

Pascal

```
procedure DrawBackground; override;
```

C++

```
virtual __fastcall DrawBackground();
```

Description

This is DrawBackground, a member of class TCustomRibbonQuickAccessPopupToolbar.

62.1.8.3.4 TCustomRibbonQuickAccessPopupToolbar.GetControlClass Method

Pascal

```
function GetControlClass(AnItem: TActionClientItem): TCustomActionControlClass; override;
```

C++

```
virtual __fastcall TCustomActionControlClass GetControlClass(TActionClientItem AnItem);
```

Description

This is GetControlClass, a member of class TCustomRibbonQuickAccessPopupToolbar.

62.1.8.3.5 TCustomRibbonQuickAccessPopupToolbar.NCPaint Method

Pascal

```
procedure NCPaint(DC: HDC); override;
```

C++

```
virtual __fastcall NCPaint(HDC DC);
```

Description

This is NCPaint, a member of class TCustomRibbonQuickAccessPopupToolbar.

62.1.8.3.6 TCustomRibbonQuickAccessPopupToolbar.ValidOrientations Method

Pascal

```
function ValidOrientations: TBarOrientations; override;
```

C++

```
virtual __fastcall TBarOrientations ValidOrientations();
```

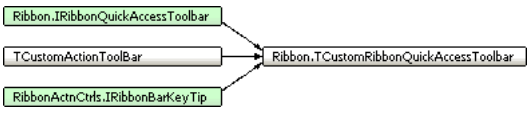
Description

This is ValidOrientations, a member of class TCustomRibbonQuickAccessPopupToolbar.

62.1.9 TCustomRibbonQuickAccessToolbar Class

Quck Access Toolbar base class.

Class Hierarchy



Pascal

```
TCustomRibbonQuickAccessToolbar = class(TCustomActionToolBar, IRibbonQuickAccessToolbar,
IRibbonBarKeyTip);
```

C++

```
class TCustomRibbonQuickAccessToolbar : public TCustomActionToolBar, public
IRibbonQuickAccessToolbar, public IRibbonBarKeyTip;
```

File

Ribbon

Description

TCustomRibbonQuickAccessToolbar is the base class Quck Access Toolbar.

See Also

Creating an Application that Uses Ribbon Controls

Adding Commands to the Ribbon

TRibbonQuickAccessToolbar (see page 3546)

62.1.9.1 TCustomRibbonQuickAccessToolbar Members

The following tables list the members exposed by TCustomRibbonQuickAccessToolbar.

Protected Methods

	Name	Description
	CMFontchanged (see page 3445)	This is CMFontchanged, a member of class TCustomRibbonQuickAccessToolbar.
V	CreateControls (see page 3446)	This is CreateControls, a member of class TCustomRibbonQuickAccessToolbar.
V	DoContextPopup (see page 3446)	OnContextPopup event dispatcher.
V	DoDropActions (see page 3446)	This is DoDropActions, a member of class TCustomRibbonQuickAccessToolbar.
V	DoDropItem (see page 3447)	This is DoDropItem, a member of class TCustomRibbonQuickAccessToolbar.
V	DoMoreCommands (see page 3447)	This is DoMoreCommands, a member of class TCustomRibbonQuickAccessToolbar.
V	DoScrollBtnClick (see page 3447)	This is DoScrollBtnClick, a member of class TCustomRibbonQuickAccessToolbar.
V	DrawBackground (see page 3448)	This is DrawBackground, a member of class TCustomRibbonQuickAccessToolbar.
V	GetBarHeight (see page 3448)	This is GetBarHeight, a member of class TCustomRibbonQuickAccessToolbar.
V	GetBarWidth (see page 3449)	This is GetBarWidth, a member of class TCustomRibbonQuickAccessToolbar.
V	GetScrollBtnClass (see page 3449)	This is GetScrollBtnClass, a member of class TCustomRibbonQuickAccessToolbar.
V	Notification (see page 3449)	Responds to notifications that components are being created or destroyed.
V	RequestAlign (see page 3450)	Instructs the parent of a control to reposition the control, enforcing its Align property.
V	SetParent (see page 3450)	Sets the parent of the control.
V	SetupDropDownBtn (see page 3451)	This is SetupDropDownBtn, a member of class TCustomRibbonQuickAccessToolbar.
V	ShowCustomizeMenu (see page 3451)	This is ShowCustomizeMenu, a member of class TCustomRibbonQuickAccessToolbar.
V	ShowMoreControls (see page 3451)	This is ShowMoreControls, a member of class TCustomRibbonQuickAccessToolbar.
V	UpdateQATWidth (see page 3451)	This is UpdateQATWidth, a member of class TCustomRibbonQuickAccessToolbar.

Public Methods

	Name	Description
	BarHandleKeyTips (see page 3444)	Check if any commands on Action Bar are partial or full match for **KeyTip**.
	BarHideKeyTips (see page 3444)	Hide any visible **KeyTip** windows.
	BarShowKeyTips (see page 3445)	Show **KeyTip**s for any commands that support them.
V	Create (see page 3443)	Creates and initializes a TCustomActionToolBar instance.

	CreateControl (see page 3445)	This is CreateControl, a member of class TCustomRibbonQuickAccessToolbar.
	Customize (see page 3446)	This is Customize, a member of class TCustomRibbonQuickAccessToolbar.
	DoDropCategory (see page 3447)	This is DoDropCategory, a member of class TCustomRibbonQuickAccessToolbar.
	DoUpdateQATWidth (see page 3448)	This is DoUpdateQATWidth, a member of class TCustomRibbonQuickAccessToolbar.
	DragDrop (see page 3448)	Generates an OnDragDrop event.

Public Nested Types

Name	Description
TMoreCommandsEvent (see page 3453)	This is nested type Ribbon.TCustomRibbonQuickAccessToolbar.TMoreCommandsEvent.

Public Properties

	Name	Description
	Items (see page 3452)	Lists the action clients that correspond to the child controls of this action band.
	OnMoreCommands (see page 3452)	This is OnMoreCommands, a member of class TCustomRibbonQuickAccessToolbar.
R	Ribbon (see page 3452)	This is Ribbon, a member of class TCustomRibbonQuickAccessToolbar.
	ScrollMode (see page 3453)	This is ScrollMode, a member of class TCustomRibbonQuickAccessToolbar.
	ShowBorder (see page 3453)	This is ShowBorder, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.2 TCustomRibbonQuickAccessToolbar.Create Constructor

Creates and initializes a TCustomActionToolBar instance.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomRibbonQuickAccessToolbar(TComponent * AOwner);
```

Description

Create generates a TCustomActionToolBar and initializes its properties.

```
__fastcall virtual TCustomActionToolBar (Classes::TComponentTComponent_Object* AOwner);
__fastcall (HWND ParentWindow) : Controls::TWinControlTWinControl_Create(ParentWindow) { }
```

62.1.9.3 TCustomRibbonQuickAccessToolbar Methods

The methods of the TCustomRibbonQuickAccessToolbar class are listed here.

Protected Methods

	Name	Description
	CMFontchanged (see page 3445)	This is CMFontchanged, a member of class TCustomRibbonQuickAccessToolbar.
	CreateControls (see page 3446)	This is CreateControls, a member of class TCustomRibbonQuickAccessToolbar.
	DoContextPopup (see page 3446)	OnContextPopup event dispatcher.
	DoDropActions (see page 3446)	This is DoDropActions, a member of class TCustomRibbonQuickAccessToolbar.
	DoDropItem (see page 3447)	This is DoDropItem, a member of class TCustomRibbonQuickAccessToolbar.
	DoMoreCommands (see page 3447)	This is DoMoreCommands, a member of class TCustomRibbonQuickAccessToolbar.
	DoScrollBtnClick (see page 3447)	This is DoScrollBtnClick, a member of class TCustomRibbonQuickAccessToolbar.
	DrawBackground (see page 3448)	This is DrawBackground, a member of class TCustomRibbonQuickAccessToolbar.
	GetBarHeight (see page 3448)	This is GetBarHeight, a member of class TCustomRibbonQuickAccessToolbar.
	GetBarWidth (see page 3449)	This is GetBarWidth, a member of class TCustomRibbonQuickAccessToolbar.

	GetScrollBtnClass (see page 3449)	This is GetScrollBtnClass, a member of class TCustomRibbonQuickAccessToolbar.
	Notification (see page 3449)	Responds to notifications that components are being created or destroyed.
	RequestAlign (see page 3450)	Instructs the parent of a control to reposition the control, enforcing its Align property.
	SetParent (see page 3450)	Sets the parent of the control.
	SetupDropDownBtn (see page 3451)	This is SetupDropDownBtn, a member of class TCustomRibbonQuickAccessToolbar.
	ShowCustomizeMenu (see page 3451)	This is ShowCustomizeMenu, a member of class TCustomRibbonQuickAccessToolbar.
	ShowMoreControls (see page 3451)	This is ShowMoreControls, a member of class TCustomRibbonQuickAccessToolbar.
	UpdateQATWidth (see page 3451)	This is UpdateQATWidth, a member of class TCustomRibbonQuickAccessToolbar.

Public Methods

	Name	Description
	BarHandleKeyTips (see page 3444)	Check if any commands on Action Bar are partial or full match for **KeyTip**.
	BarHideKeyTips (see page 3444)	Hide any visible **KeyTip** windows.
	BarShowKeyTips (see page 3445)	Show **KeyTip**s for any commands that support them.
	CreateControl (see page 3445)	This is CreateControl, a member of class TCustomRibbonQuickAccessToolbar.
	Customize (see page 3446)	This is Customize, a member of class TCustomRibbonQuickAccessToolbar.
	DoDropCategory (see page 3447)	This is DoDropCategory, a member of class TCustomRibbonQuickAccessToolbar.
	DoUpdateQATWidth (see page 3448)	This is DoUpdateQATWidth, a member of class TCustomRibbonQuickAccessToolbar.
	DragDrop (see page 3448)	Generates an OnDragDrop event.

62.1.9.3.1 TCustomRibbonQuickAccessToolbar.BarHandleKeyTips Method

Check if any commands on Action Bar are partial or full match for **KeyTip**.

Pascal

```
procedure BarHandleKeyTips(const KeyTip: string; var KeyTips: TKeyTipList);
```

C++

```
__fastcall BarHandleKeyTips(const UnicodeString KeyTip, TKeyTipList KeyTips);
```

Description

BarHandleKeyTips checks if any commands on the Action Bar are a partial or full match for the **KeyTip**. Matches are returned as a TKeyTipMatchItem in the `List` parameter. If only one item is a match, that command is executed.

See Also

TKeyTipMatchItem (see page 3598)

BarHideKeyTips (see page 3913)

BarShowKeyTips (see page 3914)

62.1.9.3.2 TCustomRibbonQuickAccessToolbar.BarHideKeyTips Method

Hide any visible **KeyTip** windows.

Pascal

```
procedure BarHideKeyTips(KeyTips: TKeyTipList);
```

C++

```
__fastcall BarHideKeyTips(TKeyTipList KeyTips);
```

Description

BarHideKeyTips hides any visible **KeyTip** windows. `ExceptList` may contain a list of TKeyTipMatchItem objects for commands that should not be hidden. These items are current partial matches and require additional handling.

See Also

TKeyTipMatchItem (see page 3598)

BarHandleKeyTips (see page 3913)

BarShowKeyTips (see page 3914)

62.1.9.3.3 TCustomRibbonQuickAccessToolbar.BarShowKeyTips Method

Show **KeyTip**s for any commands that support them.

Pascal

```
procedure BarShowKeyTips;
```

C++

```
__fastcall BarShowKeyTips();
```

Description

BarShowKeyTips shows **KeyTip**s for any commands that support them.

See Also

BarHandleKeyTips (see page 3913)

BarHideKeyTips (see page 3913)

62.1.9.3.4 TCustomRibbonQuickAccessToolbar.CMFontchanged Method

Pascal

```
procedure CMFontchanged(var Message: TMessage); message CM_FONTCHANGED;
```

C++

```
__fastcall CMFontchanged(TMessage Message);
```

Description

This is CMFontchanged, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.5 TCustomRibbonQuickAccessToolbar.CreateControl Method

Pascal

```
function CreateControl(AnItem: TActionClientItem): TCustomActionControl; override;
```

C++

```
virtual __fastcall TCustomActionControl CreateControl(TActionClientItem AnItem);
```

Description

This is CreateControl, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.6 TCustomRibbonQuickAccessToolbar.CreateControls Method

Pascal

```
procedure CreateControls; override;
```

C++

```
virtual __fastcall CreateControls();
```

Description

This is CreateControls, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.7 TCustomRibbonQuickAccessToolbar.Customize Method

Pascal

```
procedure Customize;
```

C++

```
__fastcall Customize();
```

Description

This is Customize, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.8 TCustomRibbonQuickAccessToolbar.DoContextPopup Method

OnContextPopup event dispatcher.

Pascal

```
procedure DoContextPopup(MousePos: TPoint; var Handled: Boolean); override;
```

C++

```
virtual __fastcall DoContextPopup(TPoint MousePos, Boolean Handled);
```

Description

OnContextPopup calls any event handler attached to the OnContextPopup event. When creating a component that descends from TControl, override OnContextPopup to provide other responses in addition to the inherited event-handler call when the user invokes the control's context menu.

MousePos indicates the position where the user right-clicked the control. If the user invoked the context menu using the keyboard, MousePos is (-1,-1).

Handled returns a Boolean value indicating whether the control should continue with its default response to the user action.

See Also

OnContextPopup

62.1.9.3.9 TCustomRibbonQuickAccessToolbar.DoDropActions Method

Pascal

```
procedure DoDropActions(Source: TActionDragObject; const X: Integer; const Y: Integer);
override;
```

C++

```
virtual __fastcall DoDropActions(TActionDragObject Source, const int X, const int Y);
```

Description

This is DoDropActions, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.10 TCustomRibbonQuickAccessToolbar.DoDropCategory Method

Pascal

```
procedure DoDropCategory(Source: TCategoryDragObject; const X: Integer; const Y: Integer);
override;
```

C++

```
virtual __fastcall DoDropCategory(TCategoryDragObject Source, const int X, const int Y);
```

Description

This is DoDropCategory, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.11 TCustomRibbonQuickAccessToolbar.DoDropItem Method

Pascal

```
procedure DoDropItem(Source: TActionItemDragObject; const X: Integer; const Y: Integer);
override;
```

C++

```
virtual __fastcall DoDropItem(TActionItemDragObject Source, const int X, const int Y);
```

Description

This is DoDropItem, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.12 TCustomRibbonQuickAccessToolbar.DoMoreCommands Method

Pascal

```
procedure DoMoreCommands; virtual;
```

C++

```
virtual __fastcall DoMoreCommands();
```

Description

This is DoMoreCommands, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.13 TCustomRibbonQuickAccessToolbar.DoScrollBtnClick Method

Pascal

```
procedure DoScrollBtnClick; override;
```

C++

```
virtual __fastcall DoScrollBtnClick();
```

Description

This is DoScrollBtnClick, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.14 TCustomRibbonQuickAccessToolbar.DoUpdateQATWidth Method

Pascal

```
procedure DoUpdateQATWidth;
```

C++

```
__fastcall DoUpdateQATWidth();
```

Description

This is DoUpdateQATWidth, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.15 TCustomRibbonQuickAccessToolbar.DragDrop Method

Generates an OnDragDrop event.

Pascal

```
procedure DragDrop(Source: TObject; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall DragDrop(TObject * Source, int X, int Y);
```

Description

DragDrop is called automatically when the user drags and drops a control onto the tool bar. This method generates an OnDragDrop event, and then updates its list of hidden controls to reflect the presence of the new child control.

Source is the dragged control that is dropped onto the tool bar.

X and Y are the mouse coordinates where Source was dropped.

See Also

DragCanceled

OnDragDrop

HiddenCount

62.1.9.3.16 TCustomRibbonQuickAccessToolbar.DrawBackground Method

Pascal

```
procedure DrawBackground; override;
```

C++

```
virtual __fastcall DrawBackground();
```

Description

This is DrawBackground, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.17 TCustomRibbonQuickAccessToolbar.GetBarHeight Method

Pascal

```
function GetBarHeight: Integer; override;
```

C++

```
virtual __fastcall int GetBarHeight();
```

Description

This is GetBarHeight, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.18 TCustomRibbonQuickAccessToolbar.GetBarWidth Method

Pascal

```
function GetBarWidth: Integer; override;
```

C++

```
virtual __fastcall int GetBarWidth();
```

Description

This is GetBarWidth, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.19 TCustomRibbonQuickAccessToolbar.GetScrollBtnClass Method

Pascal

```
function GetScrollBtnClass: TCustomToolScrollBtnClass; override;
```

C++

```
virtual __fastcall TCustomToolScrollBtnClass GetScrollBtnClass();
```

Description

This is GetScrollBtnClass, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.20 TCustomRibbonQuickAccessToolbar.Notification Method

Responds to notifications that components are being created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification allows TControl to update its PopupMenu property when the pop-up menu it refers to is destroyed.

Override the Notification method to respond to notifications that other components are about to be destroyed or have just been created. Use the Notification method to update controls that rely on other objects. Remove references to objects that are going away so that the control does not refer to objects that have been destroyed. For example, data-aware objects override the Notification method to update themselves when their data source is removed. Some objects respond to notifications that other objects of a specific type are being created. For example, the session component sets the Session property of new data-aware controls to itself when the AutoSessionName property is true.

By default, components pass along the notification to their owned components, if any.

See Also

PopupMenu

Notification

62.1.9.3.21 TCustomRibbonQuickAccessToolbar.RequestAlign Method

Instructs the parent of a control to reposition the control, enforcing its Align property.

Pascal

```
procedure RequestAlign; override;
```

C++

```
virtual __fastcall RequestAlign();
```

Description

Controls call RequestAlign internally when changes are made to the size or position of the control, so that they remain properly aligned within the parent control. Do not call RequestAlign in application code.

See Also

Align

Parent

62.1.9.3.22 TCustomRibbonQuickAccessToolbar.SetParent Method

Sets the parent of the control.

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

SetParent is the protected implementation of the Parent property. Override SetParent to execute additional code when the value of the Parent property changes.

The AParent parameter specifies the new parent of the control.

If the control already has a parent, SetParent removes the control from that parent's list of controls by calling the parent's RemoveControl method. If AParent is not nil (Delphi) or NULL (C++), SetParent calls its InsertControl method to add the control to its list of controls.

The TControl destructor calls SetParent(nil (Delphi) or NULL (C++)) to remove the control from its parent's control list before destroying the component.

See Also

Destroy

InsertControl

Parent

RemoveControl

62.1.9.3.23 TCustomRibbonQuickAccessToolbar.SetupDropDownBtn Method

Pascal

```
procedure SetupDropDownBtn; override;
```

C++

```
virtual __fastcall SetupDropDownBtn();
```

Description

This is SetupDropDownBtn, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.24 TCustomRibbonQuickAccessToolbar.ShowCustomizeMenu Method

Pascal

```
procedure ShowCustomizeMenu(AScrollBtn: TRibbonQATScrollButton); virtual;
```

C++

```
virtual __fastcall ShowCustomizeMenu(TRibbonQATScrollButton AScrollBtn);
```

Description

This is ShowCustomizeMenu, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.25 TCustomRibbonQuickAccessToolbar.ShowMoreControls Method

Pascal

```
procedure ShowMoreControls; virtual;
```

C++

```
virtual __fastcall ShowMoreControls();
```

Description

This is ShowMoreControls, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.3.26 TCustomRibbonQuickAccessToolbar.UpdateQATWidth Method

Pascal

```
procedure UpdateQATWidth; virtual;
```

C++

```
virtual __fastcall UpdateQATWidth();
```

Description

This is UpdateQATWidth, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.4 TCustomRibbonQuickAccessToolbar Properties

The properties of the TCustomRibbonQuickAccessToolbar class are listed here.

Public Properties

	Name	Description
	Items (see page 3452)	Lists the action clients that correspond to the child controls of this action band.
	OnMoreCommands (see page 3452)	This is OnMoreCommands, a member of class TCustomRibbonQuickAccessToolbar.
R	Ribbon (see page 3452)	This is Ribbon, a member of class TCustomRibbonQuickAccessToolbar.
	ScrollMode (see page 3453)	This is ScrollMode, a member of class TCustomRibbonQuickAccessToolbar.
	ShowBorder (see page 3453)	This is ShowBorder, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.4.1 TCustomRibbonQuickAccessToolbar.Items Property

Lists the action clients that correspond to the child controls of this action band.

Pascal

```
property Items;
```

C++

```
__property Items;
```

Description

Items is used internally to access the components that the action manager lists to represent the child controls of this action band. Each action client object in the Items collection corresponds to one of the controls that the ActionControls property lists. The entire collection of items belongs to the action client specified by this action band's ActionClient property.

See Also

ActionControls

ActionClient

ContextBar

62.1.9.4.2 TCustomRibbonQuickAccessToolbar.OnMoreCommands Property

Pascal

```
property OnMoreCommands: TMoreCommandsEvent;
```

C++

```
__property TMoreCommandsEvent OnMoreCommands;
```

Description

This is OnMoreCommands, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.4.3 TCustomRibbonQuickAccessToolbar.Ribbon Property

Pascal

```
property Ribbon: TCustomRibbon;
```

C++

```
__property TCustomRibbon Ribbon;
```

Description

This is Ribbon, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.4.4 TCustomRibbonQuickAccessToolbar.ScrollMode Property

Pascal

```
property ScrollMode: TQATScrollMode;
```

C++

```
__property TQATScrollMode ScrollMode;
```

Description

This is ScrollMode, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.4.5 TCustomRibbonQuickAccessToolbar.ShowBorder Property

Pascal

```
property ShowBorder: Boolean;
```

C++

```
__property Boolean ShowBorder;
```

Description

This is ShowBorder, a member of class TCustomRibbonQuickAccessToolbar.

62.1.9.5 TCustomRibbonQuickAccessToolbar Nested Types

The nested types of the TCustomRibbonQuickAccessToolbar class are listed here.

Public Nested Types

Name	Description
TMoreCommandsEvent (see page 3453)	This is nested type Ribbon.TCustomRibbonQuickAccessToolbar.TMoreCommandsEvent.

62.1.9.5.1 TCustomRibbonQuickAccessToolbar.TMoreCommandsEvent Nested Type

Pascal

```
TMoreCommandsEvent = procedure (Sender: TObject; var Handled: Boolean) of object;
```

C++

```
(Sender: TObject; var Handled: Boolean) ( TMoreCommandsEvent)();
```

Description

This is nested type Ribbon.TCustomRibbonQuickAccessToolbar.TMoreCommandsEvent.

62.1.10 TCustomTabData Class

Encapsulates all functions and status to manage scrolling of tabs and selected group.

Class Hierarchy

Pascal

```
TCustomTabData = class;
```

C++

```
class TCustomTabData;
```

File

Ribbon

Description

TCustomTabData encapsulates all the functions and status information needed to manage scrolling tabs and the selected group on a ribbon.

62.1.10.1 TCustomTabData Members

The following tables list the members exposed by TCustomTabData.

Public Methods

	Name	Description
	Add (see page 3455)	This is Add, a member of class TCustomTabData.
	CalculateTabWidths (see page 3455)	This is CalculateTabWidths, a member of class TCustomTabData.
	Clear (see page 3456)	This is Clear, a member of class TCustomTabData.
V	Create (see page 3454)	This is Create, a member of class TCustomTabData.
V	Destroy (see page 3455)	This is Destroy, a member of class TCustomTabData.

Public Properties

	Name	Description
R	AvailableSpace (see page 3456)	This is AvailableSpace, a member of class TCustomTabData.
	FirstVisibleTabIndex (see page 3456)	This is FirstVisibleTabIndex, a member of class TCustomTabData.
R	FullWidth (see page 3457)	This is FullWidth, a member of class TCustomTabData.
R	ItemCount (see page 3457)	This is ItemCount, a member of class TCustomTabData.
R	LastVisibleTabIndex (see page 3457)	This is LastVisibleTabIndex, a member of class TCustomTabData.
R	LeftTabScrollButtonRect (see page 3457)	This is LeftTabScrollButtonRect, a member of class TCustomTabData.
R	NewWidth (see page 3458)	This is NewWidth, a member of class TCustomTabData.
R	Owner (see page 3458)	This is Owner, a member of class TCustomTabData.
R	RightTabScrollButtonRect (see page 3458)	This is RightTabScrollButtonRect, a member of class TCustomTabData.
R	ScrollButtons (see page 3458)	This is ScrollButtons, a member of class TCustomTabData.
R	SeparatorOpacity (see page 3458)	This is SeparatorOpacity, a member of class TCustomTabData.
R	TabSeparatorState (see page 3459)	This is TabSeparatorState, a member of class TCustomTabData.
R	TabUpdateRect (see page 3459)	This is TabUpdateRect, a member of class TCustomTabData.

62.1.10.2 TCustomTabData.Create Constructor

Pascal

```
constructor Create(AOwner: TCustomRibbon); virtual; reintroduce;
```

C++

```
virtual __fastcall TCustomTabData(TCustomRibbon AOwner);
```

Description

This is Create, a member of class TCustomTabData.

62.1.10.3 TCustomTabData.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomTabData();
```

Description

This is Destroy, a member of class TCustomTabData.

62.1.10.4 TCustomTabData Methods

The methods of the TCustomTabData class are listed here.

Public Methods

	Name	Description
	Add (see page 3455)	This is Add, a member of class TCustomTabData.
	CalculateTabWidths (see page 3455)	This is CalculateTabWidths, a member of class TCustomTabData.
	Clear (see page 3456)	This is Clear, a member of class TCustomTabData.

62.1.10.4.1 TCustomTabData.Add Method

Pascal

```
procedure Add(ATab: TRibbonTabItem);
```

C++

```
__fastcall Add(TRibbonTabItem ATab);
```

Description

This is Add, a member of class TCustomTabData.

62.1.10.4.2 TCustomTabData.CalculateTabWidths Method

Pascal

```
procedure CalculateTabWidths(AAvailableSpace: Integer; APages: TRibbonTabs);
```

C++

```
__fastcall CalculateTabWidths(int AAvailableSpace, TRibbonTabs APages);
```

Description

This is CalculateTabWidths, a member of class TCustomTabData.

62.1.10.4.3 TCustomTabData.Clear Method

Pascal

```
procedure Clear;
```

C++

```
__fastcall Clear();
```

Description

This is Clear, a member of class TCustomTabData.

62.1.10.5 TCustomTabData Properties

The properties of the TCustomTabData class are listed here.

Public Properties

	Name	Description
R	AvailableSpace (see page 3456)	This is AvailableSpace, a member of class TCustomTabData.
	FirstVisibleTabIndex (see page 3456)	This is FirstVisibleTabIndex, a member of class TCustomTabData.
R	FullWidth (see page 3457)	This is FullWidth, a member of class TCustomTabData.
R	ItemCount (see page 3457)	This is ItemCount, a member of class TCustomTabData.
R	LastVisibleTabIndex (see page 3457)	This is LastVisibleTabIndex, a member of class TCustomTabData.
R	LeftTabScrollButtonRect (see page 3457)	This is LeftTabScrollButtonRect, a member of class TCustomTabData.
R	NewWidth (see page 3458)	This is NewWidth, a member of class TCustomTabData.
R	Owner (see page 3458)	This is Owner, a member of class TCustomTabData.
R	RightTabScrollButtonRect (see page 3458)	This is RightTabScrollButtonRect, a member of class TCustomTabData.
R	ScrollButtons (see page 3458)	This is ScrollButtons, a member of class TCustomTabData.
R	SeparatorOpacity (see page 3458)	This is SeparatorOpacity, a member of class TCustomTabData.
R	TabSeparatorState (see page 3459)	This is TabSeparatorState, a member of class TCustomTabData.
R	TabUpdateRect (see page 3459)	This is TabUpdateRect, a member of class TCustomTabData.

62.1.10.5.1 TCustomTabData.AvailableSpace Property

Pascal

```
property AvailableSpace: Integer;
```

C++

```
__property int AvailableSpace;
```

Description

This is AvailableSpace, a member of class TCustomTabData.

62.1.10.5.2 TCustomTabData.FirstVisibleTabIndex Property

Pascal

```
property FirstVisibleTabIndex: Integer;
```

C++

```
__property int FirstVisibleTabIndex;
```

Description

This is FirstVisibleTabIndex, a member of class TCustomTabData.

62.1.10.5.3 TCustomTabData.FullWidth Property

Pascal

```
property FullWidth [Idx: Integer]: Integer;
```

C++

```
__property int FullWidth[int Idx];
```

Description

This is FullWidth, a member of class TCustomTabData.

62.1.10.5.4 TCustomTabData.ItemCount Property

Pascal

```
property ItemCount: Integer;
```

C++

```
__property int ItemCount;
```

Description

This is ItemCount, a member of class TCustomTabData.

62.1.10.5.5 TCustomTabData.LastVisibleTabIndex Property

Pascal

```
property LastVisibleTabIndex: Integer;
```

C++

```
__property int LastVisibleTabIndex;
```

Description

This is LastVisibleTabIndex, a member of class TCustomTabData.

62.1.10.5.6 TCustomTabData.LeftTabScrollButtonRect Property

Pascal

```
property LeftTabScrollButtonRect: TRect;
```

C++

```
__property TRect LeftTabScrollButtonRect;
```

Description

This is LeftTabScrollButtonRect, a member of class TCustomTabData.

62.1.10.5.7 TCustomTabData.NewWidth Property

Pascal

```
property NewWidth [Idx: Integer]: Integer;
```

C++

```
__property int NewWidth[int Idx];
```

Description

This is NewWidth, a member of class TCustomTabData.

62.1.10.5.8 TCustomTabData.Owner Property

Pascal

```
property Owner: TCustomRibbon;
```

C++

```
__property TCustomRibbon Owner;
```

Description

This is Owner, a member of class TCustomTabData.

62.1.10.5.9 TCustomTabData.RightTabScrollButtonRect Property

Pascal

```
property RightTabScrollButtonRect: TRect;
```

C++

```
__property TRect RightTabScrollButtonRect;
```

Description

This is RightTabScrollButtonRect, a member of class TCustomTabData.

62.1.10.5.10 TCustomTabData.ScrollButtons Property

Pascal

```
property ScrollButtons: TRibbonScrollButtons;
```

C++

```
__property TRibbonScrollButtons ScrollButtons;
```

Description

This is ScrollButtons, a member of class TCustomTabData.

62.1.10.5.11 TCustomTabData.SeparatorOpacity Property

Pascal

```
property SeparatorOpacity: Byte;
```

C++

```
__property Byte SeparatorOpacity;
```

Description

This is SeparatorOpacity, a member of class TCustomTabData.

62.1.10.5.12 TCustomTabData.TabSeparatorState Property

Pascal

```
property TabSeparatorState: TTabSeparatorState;
```

C++

```
__property TTabSeparatorState TabSeparatorState;
```

Description

This is TabSeparatorState, a member of class TCustomTabData.

62.1.10.5.13 TCustomTabData.TabUpdateRect Property

Pascal

```
property TabUpdateRect: TRect;
```

C++

```
__property TRect TabUpdateRect;
```

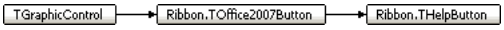
Description

This is TabUpdateRect, a member of class TCustomTabData.

62.1.11 THelpButton Class

Button to activate help.

Class Hierarchy



Pascal

```
THelpButton = class(TOffice2007Button);
```

C++

```
class THelpButton : public TOffice2007Button;
```

File

Ribbon

Description

THelpButton represents the button used for activating help.

See Also

ShowHelpButton (see page 3398)

62.1.11.1 THelpButton Members

The following tables list the members exposed by THelpButton.

Protected Methods

	Name	Description
	CMMouseenter (see page 3463)	This is CMMouseenter, a member of class TOffice2007Button.
	CMMouseleave (see page 3463)	This is CMMouseleave, a member of class TOffice2007Button.
V	DoClick (see page 3463)	This is DoClick, a member of class TOffice2007Button.
V	DrawBackground (see page 3464)	This is DrawBackground, a member of class TOffice2007Button.
V	DrawImage (see page 3460)	This is DrawImage, a member of class THelpButton.
V	MouseDown (see page 3464)	OnMouseDown event dispatcher.
V	MouseMove (see page 3465)	Respond to mouse moving over control area..
V	MouseUp (see page 3465)	OnMouseUp event dispatcher.
V	Paint (see page 3466)	Render control surface.
V	SetParent (see page 3466)	Sets the parent of the control.

Protected Properties

	Name	Description
	MouseInControl (see page 3467)	This is MouseInControl, a member of class TOffice2007Button.
R	RibbonStyle (see page 3467)	This is RibbonStyle, a member of class TOffice2007Button.

Public Events

	Name	Description
	OnClick (see page 3467)	Occurs when the user clicks the control.

Public Methods

	Name	Description
V	Create (see page 3462)	Creates an instance of TGraphicControl.
V	Destroy (see page 3462)	Destroys an instance of TGraphicControl.

62.1.11.2 THelpButton Methods

The methods of the THelpButton class are listed here.

Protected Methods

	Name	Description
V	DrawImage (see page 3460)	This is DrawImage, a member of class THelpButton.

62.1.11.2.1 THelpButton.DrawImage Method

Pascal

```
procedure DrawImage; override;
```

C++

```
virtual __fastcall DrawImage();
```

Description

This is DrawImage, a member of class THelpButton.

62.1.12 TOffice2007Button Class

Class Hierarchy

Pascal

```
TOffice2007Button = class(TGraphicControl);
```

C++

```
class TOffice2007Button : public TGraphicControl;
```

File

Ribbon

Description

This is class Ribbon.TOffice2007Button.

62.1.12.1 TOffice2007Button Members

The following tables list the members exposed by TOffice2007Button.

Protected Methods

	Name	Description
	CMMouseenter (see page 3463)	This is CMMouseenter, a member of class TOffice2007Button.
	CMMouseleave (see page 3463)	This is CMMouseleave, a member of class TOffice2007Button.
V	DoClick (see page 3463)	This is DoClick, a member of class TOffice2007Button.
V	DrawBackground (see page 3464)	This is DrawBackground, a member of class TOffice2007Button.
V	DrawImage (see page 3464)	This is DrawImage, a member of class TOffice2007Button.
V	MouseDown (see page 3464)	OnMouseDown event dispatcher.
V	MouseMove (see page 3465)	Respond to mouse moving over control area..
V	MouseUp (see page 3465)	OnMouseUp event dispatcher.
V	Paint (see page 3466)	Render control surface.
V	SetParent (see page 3466)	Sets the parent of the control.

Protected Properties

	Name	Description
	MouseInControl (see page 3467)	This is MouseInControl, a member of class TOffice2007Button.
R	RibbonStyle (see page 3467)	This is RibbonStyle, a member of class TOffice2007Button.

Public Events

	Name	Description
	OnClick (see page 3467)	Occurs when the user clicks the control.

Public Methods

	Name	Description
V	Create (see page 3462)	Creates an instance of TGraphicControl.
V	Destroy (see page 3462)	Destroys an instance of TGraphicControl.

62.1.12.2 TOffice2007Button.Create Constructor

Creates an instance of TGraphicControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TOffice2007Button(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TGraphicControl. However, you should never attempt to instantiate a TGraphicControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

AOwner is the component, typically a form, that is responsible for freeing the graphic control. It becomes the value of the Owner property. If you do not explicitly provide an Owner to the constructor (that is, if AOwner is nil (Delphi) or NULL (C++)), your application is responsible for explicitly freeing the graphic control.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Owner

Create

Destroy

62.1.12.3 TOffice2007Button.Destroy Destructor

Destroys an instance of TGraphicControl.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TOffice2007Button();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the graphic control reference is not nil, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

As the control is destroyed, it:

Frees the TCanvas object in its Canvas property.

Override Destroy to free any memory or resources allocated in the Create method. When declaring a Destroy method in a descendant control type, always add the override directive to the declaration and call the inherited Destroy as the last statement in the redeclared method.

When a control is created, Delphi automatically provides exception handling around the constructor call. The destructor is called if an exception escapes from the constructor. This means that the destructor code must be prepared to clean up a partially constructed instance. Check all data fields for zero before disposing of their contents.

See Also

Create

System::TObject::Free

62.1.12.4 TOffice2007Button Methods

The methods of the TOffice2007Button class are listed here.

Protected Methods

	Name	Description
	CMMouseenter (see page 3463)	This is CMMouseenter, a member of class TOffice2007Button.
	CMMouseleave (see page 3463)	This is CMMouseleave, a member of class TOffice2007Button.
V	DoClick (see page 3463)	This is DoClick, a member of class TOffice2007Button.
V	DrawBackground (see page 3464)	This is DrawBackground, a member of class TOffice2007Button.
V	DrawImage (see page 3464)	This is DrawImage, a member of class TOffice2007Button.
V	MouseDown (see page 3464)	OnMouseDown event dispatcher.
V	MouseMove (see page 3465)	Respond to mouse moving over control area..
V	MouseUp (see page 3465)	OnMouseUp event dispatcher.
V	Paint (see page 3466)	Render control surface.
V	SetParent (see page 3466)	Sets the parent of the control.

62.1.12.4.1 TOffice2007Button.CMMouseenter Method

Pascal

```
procedure CMMouseenter(var Message: TMessage); message CM_MOUSEENTER;
```

C++

```
__fastcall CMMouseenter(TMessage Message);
```

Description

This is CMMouseenter, a member of class TOffice2007Button.

62.1.12.4.2 TOffice2007Button.CMMouseleave Method

Pascal

```
procedure CMMouseleave(var Message: TMessage); message CM_MOUSELEAVE;
```

C++

```
__fastcall CMMouseleave(TMessage Message);
```

Description

This is CMMouseleave, a member of class TOffice2007Button.

62.1.12.4.3 TOffice2007Button.DoClick Method

Pascal

```
procedure DoClick; virtual;
```

C++

```
virtual __fastcall DoClick();
```

Description

This is DoClick, a member of class TOffice2007Button.

62.1.12.4.4 TOffice2007Button.DrawBackground Method

Pascal

```
procedure DrawBackground; virtual;
```

C++

```
virtual __fastcall DrawBackground();
```

Description

This is DrawBackground, a member of class TOffice2007Button.

62.1.12.4.5 TOffice2007Button.DrawImage Method

Pascal

```
procedure DrawImage; virtual;
```

C++

```
virtual __fastcall DrawImage();
```

Description

This is DrawImage, a member of class TOffice2007Button.

62.1.12.4.6 TOffice2007Button.MouseDown Method

OnMouseDown event dispatcher.

Pascal

```
procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer);
override;
```

C++

```
virtual __fastcall MouseDown(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user presses the mouse button down while the cursor's hotspot is over the control.

The Button parameter determines which mouse button the user pressed. Shift indicates which shift keys (Shift, Ctrl, or Alt) were down when the user pressed the mouse button. X and Y are the pixel coordinates of the mouse pointer within the client area of the control.

A control calls MouseDown in response to any of the Windows mouse-down messages (WM_LBUTTONDOWN, WM_MBUTTONDOWN, WM_RBUTTONDOWN), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was pressed down: left, right, or middle.

See Also

MouseMove

MouseUp

OnMouseDown

62.1.12.4.7 TOffice2007Button.MouseMove Method

Respond to mouse moving over control area..

Pascal

```
procedure MouseMove(Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseMove(TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseMove method to provide other responses in addition to calling the OnMouseMove event handler when the user moves the mouse.

A control calls MouseMove in response to any of the Windows mouse-move messages (WM_MOUSEMOVE), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively.

As the mouse cursor moves across a control, this method is called repeatedly. Each time it is called, it is with the new coordinates that reflect the continuous path of the mouse cursor across the screen real estate covered by the control's visual representation.

See Also

MouseDown

MouseUp

OnMouseMove

62.1.12.4.8 TOffice2007Button.MouseUp Method

OnMouseUp event dispatcher.

Pascal

```
procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseUp(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user releases a previously pressed mouse button while the cursor's hotspot is over the control.

A control calls MouseUp in response to any of the Windows mouse-up messages (WM_LBUTTONUP, WM_MBUTTONUP, WM_RBUTTONUP), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was released: left, right, or middle

See Also

MouseDown

MouseMove

OnMouseUp

62.1.12.4.9 TOffice2007Button.Paint Method

Render control surface.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method is called automatically when a lightweight control needs update its display area. Code in the Paint method uses the Canvas property to render the control surface.

As implemented in TGraphicControl, Paint does nothing. Specific rendering logic is provided by Paint methods in descendant control classes.

See Also

Canvas

62.1.12.4.10 TOffice2007Button.SetParent Method

Sets the parent of the control.

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

SetParent is the protected implementation of the Parent property. Override SetParent to execute additional code when the value of the Parent property changes.

The AParent parameter specifies the new parent of the control.

If the control already has a parent, SetParent removes the control from that parent's list of controls by calling the parent's RemoveControl method. If AParent is not nil (Delphi) or NULL (C++), SetParent calls its InsertControl method to add the control to its list of controls.

The TControl destructor calls SetParent(nil (Delphi) or NULL (C++)) to remove the control from its parent's control list before destroying the component.

See Also

Destroy

InsertControl

Parent

RemoveControl

62.1.12.5 TOffice2007Button Properties

The properties of the TOffice2007Button class are listed here.

Protected Properties

	Name	Description
	MouseInControl (see page 3467)	This is MouseInControl, a member of class TOffice2007Button.
R	RibbonStyle (see page 3467)	This is RibbonStyle, a member of class TOffice2007Button.

62.1.12.5.1 TOffice2007Button.MouseInControl Property

Pascal

```
property MouseInControl: Boolean;
```

C++

```
__property Boolean MouseInControl;
```

Description

This is MouseInControl, a member of class TOffice2007Button.

62.1.12.5.2 TOffice2007Button.RibbonStyle Property

Pascal

```
property RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__property TRibbonStyleActionBars RibbonStyle;
```

Description

This is RibbonStyle, a member of class TOffice2007Button.

62.1.12.6 TOffice2007Button Events

The events of the TOffice2007Button class are listed here.

Public Events

	Name	Description
	OnClick (see page 3467)	Occurs when the user clicks the control.

62.1.12.6.1 TOffice2007Button.OnClick Event

Occurs when the user clicks the control.

Pascal

```
property OnClick: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnClick;
```

Description

Use the OnClick event handler to respond when the user clicks the control. If the control has an associated action, and that action has an OnExecute method, the action's OnExecute method responds to click events unless it is superseded by an OnClick event handler.

Usually OnClick occurs when the user presses and releases the left mouse button with the mouse pointer over the control. This event can also occur when:

- The user selects an item in a grid, outline, list, or combo box by pressing an arrow key.
- The user presses Spacebar while a button or check box has focus.
- The user presses Enter when the active form has a default button (specified by the Default property).
- The user presses Esc when the active form has a cancel button (specified by the Cancel property).
- The user presses the accelerator key for a button or check box. For example, if the value of the Caption property of a check box is '&Bold', the B is underlined at runtime and the OnClick event of the check box is triggered when the user presses Alt+B. However, focus does not move to the control in these instances.
- The Checked property of a radio button is set to true.
- The value of the Checked property of a check box is changed.
- The Click method of a menu item is called.

For a form, an OnClick event occurs when the user clicks a blank area of the form or on a disabled component.

C++ Examples:

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
    pPage->PageControl = PageControl1;
    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this); // The owner will clean this up.
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
```

```
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
    begin
      PageControl := PageControl1;
      Caption := 'TabSheet' + IntToStr(i);
      with TEdit.Create(Self) do
      begin
        Parent := PageControl1.Pages[i];
        Left := Random(PageControl1.ActivePage.ClientWidth - Width);
        Top := Random(PageControl1.ActivePage.ClientHeight - Height);
      end;
    end;
    PageControl1Change(Sender);
end;
```

See Also

OnDblClick

Cancel

Default

Click (see page 1375)

OnExecute

Action

Adding Silent Video Clips to an Application

62.1.13 TQuickAccessToolbar Class

Quick Access Toolbar appearance.

Class Hierarchy

Pascal

```
TQuickAccessToolbar = class(TPersistent);
```

C++

```
class TQuickAccessToolbar : public TPersistent;
```

File

Ribbon

Description

TQuickAccessToolbar defines the Quick Access Toolbar appearance. This class contains a reference to the Quick Access Toolbar on the form.

See Also

Creating an Application that Uses Ribbon Controls

Adding Commands to the Ribbon

62.1.13.1 TQuickAccessToolbar Members

The following tables list the members exposed by TQuickAccessToolbar.

Enumerations

	Name	Description
	TQATPosition (see page 3471)	Position (see page 3474) of Quick Access Toolbar.

Protected Methods

	Name	Description
	DoChange (see page 3472)	This is DoChange, a member of class TQuickAccessToolbar.

Public Events

	Name	Description
	OnChange (see page 3475)	This is OnChange, a member of class TQuickAccessToolbar.

Public Methods

	Name	Description
	Assign (see page 3472)	Copies the contents of another, similar object.
	Create (see page 3470)	This is Create, a member of class TQuickAccessToolbar.
	Show (see page 3473)	This is Show, a member of class TQuickAccessToolbar.
	TogglePosition (see page 3473)	This is TogglePosition, a member of class TQuickAccessToolbar.

Public Properties

	Name	Description
	BarWidth (see page 3474)	This is BarWidth, a member of class TQuickAccessToolbar.

Published Properties

	Name	Description
	ActionBar (see page 3473)	This is ActionBar, a member of class TQuickAccessToolbar.
	Position (see page 3474)	This is Position, a member of class TQuickAccessToolbar.
	ShowMinimize (see page 3474)	This is ShowMinimize, a member of class TQuickAccessToolbar.
	ShowMoreCommands (see page 3474)	This is ShowMoreCommands, a member of class TQuickAccessToolbar.
	ShowPosition (see page 3474)	This is ShowPosition, a member of class TQuickAccessToolbar.

62.1.13.2 TQuickAccessToolbar.Create Constructor

Pascal

```
constructor Create(ARibbon: TCustomRibbon); reintroduce;
```

C++

```
__fastcall TQuickAccessToolbar(TCustomRibbon ARibbon);
```

Description

This is Create, a member of class TQuickAccessToolbar.

62.1.13.3 TQuickAccessToolbar Enumerations

The enumerations of the TQuickAccessToolbar class are listed here.

Enumerations

	Name	Description
	TQATPosition (see page 3471)	Position (see page 3474) of Quick Access Toolbar.

62.1.13.3.1 Ribbon.TQuickAccessToolbar.TQATPosition Enumeration

Position (see page 3474) of Quick Access Toolbar.

Pascal

```
TQATPosition = (
  qpTop,
  qpBottom
);
```

C++

```
enum TQATPosition {
  qpTop,
  qpBottom
};
```

File

Ribbon

Description

TQATPosition specifies the position of the Quick Access Toolbar. It can have one of the following values:

Value	Meaning
qpTop	Quick Access Toolbar is positioned at the top of the ribbon.
gpBottom	Quick Access Toolbar is positioned at the base of the ribbon.

See Also

TQuickAccessToolbar (see page 3469)

62.1.13.4 TQuickAccessToolbar Methods

The methods of the TQuickAccessToolbar class are listed here.

Protected Methods

	Name	Description
	DoChange (see page 3472)	This is DoChange, a member of class TQuickAccessToolbar.

Public Methods

	Name	Description
	Assign (see page 3472)	Copies the contents of another, similar object.
	Show (see page 3473)	This is Show, a member of class TQuickAccessToolbar.
	TogglePosition (see page 3473)	This is TogglePosition, a member of class TQuickAccessToolbar.

62.1.13.4.1 TQuickAccessToolbar.Assign Method

Copies the contents of another, similar object.

Pascal

```
procedure Assign(Source: TPersistent); override;
```

C++

```
virtual __fastcall Assign(TPersistent * Source);
```

Description

Call Assign to copy the properties or other attributes of one object from another. The standard form of a call to Assign is

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

which tells the Destination object to copy the contents of the Source object to itself.

Most objects override Assign to handle the assignment of properties from similar objects. When overriding Assign, call the inherited method if the destination object can't handle the assignment of properties from the class of the Source parameter.

If no overridden Assign method can handle the assignment of properties from Source, the method implemented in TPersistent calls the source object's AssignTo method. This allows the source object to handle the assignment. If the Source object is nil (Delphi) or NULL (C++), Assign raises an EConvertError exception.

In general, the statement

```
Destination := Source; {Delphi}
Destination = Source; // C++
```

is not the same as the statement

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

The assignment operator makes Destination reference the same object as Source, whereas the Assign method copies the contents of the object referenced by Source into the object referenced by Destination.

Note: The types of some properties are also objects. If these properties have write methods that use Assign to set the value of the property, then in these cases the assignment operator does the same thing as the Assign method.

See Also

AssignTo

EConvertError

62.1.13.4.2 TQuickAccessToolbar.DoChange Method

Pascal

```
procedure DoChange; virtual;
```

C++

```
virtual __fastcall DoChange();
```

Description

This is DoChange, a member of class TQuickAccessToolbar.

62.1.13.4.3 TQuickAccessToolbar.Show Method

Pascal

```
procedure Show(const Visible: Boolean);
```

C++

```
__fastcall Show(const Boolean Visible);
```

Description

This is Show, a member of class TQuickAccessToolbar.

62.1.13.4.4 TQuickAccessToolbar.TogglePosition Method

Pascal

```
procedure TogglePosition;
```

C++

```
__fastcall TogglePosition();
```

Description

This is TogglePosition, a member of class TQuickAccessToolbar.

62.1.13.5 TQuickAccessToolbar Properties

The properties of the TQuickAccessToolbar class are listed here.

Public Properties

	Name	Description
	BarWidth (see page 3474)	This is BarWidth, a member of class TQuickAccessToolbar.

Published Properties

	Name	Description
	ActionBar (see page 3473)	This is ActionBar, a member of class TQuickAccessToolbar.
	Position (see page 3474)	This is Position, a member of class TQuickAccessToolbar.
	ShowMinimize (see page 3474)	This is ShowMinimize, a member of class TQuickAccessToolbar.
	ShowMoreCommands (see page 3474)	This is ShowMoreCommands, a member of class TQuickAccessToolbar.
	ShowPosition (see page 3474)	This is ShowPosition, a member of class TQuickAccessToolbar.

62.1.13.5.1 TQuickAccessToolbar.ActionBar Property

Pascal

```
property ActionBar: TCustomRibbonQuickAccessToolbar;
```

C++

```
__property TCustomRibbonQuickAccessToolbar ActionBar;
```

Description

This is ActionBar, a member of class TQuickAccessToolbar.

62.1.13.5.2 TQuickAccessToolbar.BarWidth Property

Pascal

```
property BarWidth: Integer;
```

C++

```
__property int BarWidth;
```

Description

This is BarWidth, a member of class TQuickAccessToolbar.

62.1.13.5.3 TQuickAccessToolbar.Position Property

Pascal

```
property Position: TQATPosition;
```

C++

```
__property TQATPosition Position;
```

Description

This is Position, a member of class TQuickAccessToolbar.

62.1.13.5.4 TQuickAccessToolbar.ShowMinimize Property

Pascal

```
property ShowMinimize: Boolean;
```

C++

```
__property Boolean ShowMinimize;
```

Description

This is ShowMinimize, a member of class TQuickAccessToolbar.

62.1.13.5.5 TQuickAccessToolbar.ShowMoreCommands Property

Pascal

```
property ShowMoreCommands: Boolean;
```

C++

```
__property Boolean ShowMoreCommands;
```

Description

This is ShowMoreCommands, a member of class TQuickAccessToolbar.

62.1.13.5.6 TQuickAccessToolbar.ShowPosition Property

Pascal

```
property ShowPosition: Boolean;
```

C++

```
__property Boolean ShowPosition;
```

Description

This is ShowPosition, a member of class TQuickAccessToolbar.

62.1.13.6 TQuickAccessToolbar Events

The events of the TQuickAccessToolbar class are listed here.

Public Events

	Name	Description
	OnChange (see page 3475)	This is OnChange, a member of class TQuickAccessToolbar.

62.1.13.6.1 TQuickAccessToolbar.OnChange Event

Pascal

```
property OnChange: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnChange;
```

Description

This is OnChange, a member of class TQuickAccessToolbar.

62.1.14 TQuickAccessToolbarGroupItem Class

Display Quick Access Toolbar commands to display entire Ribbon (see page 3347) group.

Class Hierarchy

Pascal

```
TQuickAccessToolbarGroupItem = class;
```

C++

```
class TQuickAccessToolbarGroupItem;
```

File

Ribbon

Description

TQuickAccessToolbarGroupItem is used to display Quick Access Toolbar commands that display an entire Ribbon (see page 3347) group when clicked.

62.1.14.1 TQuickAccessToolbarGroupItem Members

The following tables list the members exposed by TQuickAccessToolbarGroupItem.

Public Methods

	Name	Description
	Create (see page 3476)	This is Create, a member of class TQuickAccessToolbarGroupItem.
V	Destroy (see page 3476)	This is Destroy, a member of class TQuickAccessToolbarGroupItem.

Public Properties

	Name	Description
R	Action (see page 3476)	This is Action, a member of class TQuickAccessToolbarGroupItem.
R	Caption (see page 3477)	This is Caption, a member of class TQuickAccessToolbarGroupItem.
R	Group (see page 3477)	This is Group, a member of class TQuickAccessToolbarGroupItem.

62.1.14.2 TQuickAccessToolbarGroupItem.Create Constructor

Pascal

```
constructor Create(ActionList: TActionList; AGroup: TCustomRibbonGroup); reintroduce;
```

C++

```
__fastcall TQuickAccessToolbarGroupItem(TActionList ActionList, TCustomRibbonGroup AGroup);
```

Description

This is Create, a member of class TQuickAccessToolbarGroupItem.

62.1.14.3 TQuickAccessToolbarGroupItem.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TQuickAccessToolbarGroupItem();
```

Description

This is Destroy, a member of class TQuickAccessToolbarGroupItem.

62.1.14.4 TQuickAccessToolbarGroupItem Properties

The properties of the TQuickAccessToolbarGroupItem class are listed here.

Public Properties

	Name	Description
R	Action (see page 3476)	This is Action, a member of class TQuickAccessToolbarGroupItem.
R	Caption (see page 3477)	This is Caption, a member of class TQuickAccessToolbarGroupItem.
R	Group (see page 3477)	This is Group, a member of class TQuickAccessToolbarGroupItem.

62.1.14.4.1 TQuickAccessToolbarGroupItem.Action Property

Pascal

```
property Action: TAction;
```

C++

```
__property TAction Action;
```

Description

This is Action, a member of class TQuickAccessToolbarGroupItem.

62.1.14.4.2 TQuickAccessToolbarGroupItem.Caption Property

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

This is Caption, a member of class TQuickAccessToolbarGroupItem.

62.1.14.4.3 TQuickAccessToolbarGroupItem.Group Property

Pascal

```
property Group: TCustomRibbonGroup;
```

C++

```
__property TCustomRibbonGroup Group;
```

Description

This is Group, a member of class TQuickAccessToolbarGroupItem.

62.1.15 TQuickAccessToolbarGroups Class

Class Hierarchy

Pascal

```
TQuickAccessToolbarGroups = class;
```

C++

```
class TQuickAccessToolbarGroups;
```

File

Ribbon

Description

This is class Ribbon.TQuickAccessToolbarGroups.

62.1.15.1 TQuickAccessToolbarGroups Members

The following tables list the members exposed by TQuickAccessToolbarGroups.

Public Methods

	Name	Description
	Add (see page 3478)	This is Add, a member of class TQuickAccessToolbarGroups.
	Clean (see page 3479)	This is Clean, a member of class TQuickAccessToolbarGroups.
	Create (see page 3478)	This is Create, a member of class TQuickAccessToolbarGroups.
	Destroy (see page 3478)	This is Destroy, a member of class TQuickAccessToolbarGroups.
	Exists (see page 3479)	This is Exists, a member of class TQuickAccessToolbarGroups.
	Group (see page 3479)	This is Group, a member of class TQuickAccessToolbarGroups.
	Remove (see page 3479)	This is Remove, a member of class TQuickAccessToolbarGroups.

62.1.15.2 TQuickAccessToolbarGroups.Create Constructor

Pascal

```
constructor Create(Images: TCustomImageList); virtual;
```

C++

```
virtual __fastcall TQuickAccessToolbarGroups(TCustomImageList Images);
```

Description

This is Create, a member of class TQuickAccessToolbarGroups.

62.1.15.3 TQuickAccessToolbarGroups.Destroy Destructor

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TQuickAccessToolbarGroups();
```

Description

This is Destroy, a member of class TQuickAccessToolbarGroups.

62.1.15.4 TQuickAccessToolbarGroups Methods

The methods of the TQuickAccessToolbarGroups class are listed here.

Public Methods

	Name	Description
	Add (see page 3478)	This is Add, a member of class TQuickAccessToolbarGroups.
	Clean (see page 3479)	This is Clean, a member of class TQuickAccessToolbarGroups.
	Exists (see page 3479)	This is Exists, a member of class TQuickAccessToolbarGroups.
	Group (see page 3479)	This is Group, a member of class TQuickAccessToolbarGroups.
	Remove (see page 3479)	This is Remove, a member of class TQuickAccessToolbarGroups.

62.1.15.4.1 TQuickAccessToolbarGroups.Add Method

Pascal

```
function Add(AGroup: TCustomRibbonGroup): TQuickAccessToolbarGroupItem;
```

C++

```
__fastcall TQuickAccessToolbarGroupItem Add(TCustomRibbonGroup AGroup);
```

Description

This is Add, a member of class TQuickAccessToolbarGroups.

62.1.15.4.2 TQuickAccessToolbarGroups.Clean Method

Pascal

```
procedure Clean;
```

C++

```
__fastcall Clean();
```

Description

This is Clean, a member of class TQuickAccessToolbarGroups.

62.1.15.4.3 TQuickAccessToolbarGroups.Exists Method

Pascal

```
function Exists(const ACaption: string): Boolean;
```

C++

```
__fastcall Boolean Exists(const UnicodeString ACaption);
```

Description

This is Exists, a member of class TQuickAccessToolbarGroups.

62.1.15.4.4 TQuickAccessToolbarGroups.Group Method

Pascal

```
function Group(const ACaption: string): TCustomRibbonGroup;
```

C++

```
__fastcall TCustomRibbonGroup Group(const UnicodeString ACaption);
```

Description

This is Group, a member of class TQuickAccessToolbarGroups.

62.1.15.4.5 TQuickAccessToolbarGroups.Remove Method

Pascal

```
procedure Remove(const ACaption: string);
```

C++

```
__fastcall Remove(const UnicodeString ACaption);
```

Description

This is Remove, a member of class TQuickAccessToolbarGroups.

62.1.16 TRibbon Class

Implement ribbon control.

Class Hierarchy

Pascal

```
TRibbon = class(TCustomRibbon);
```

C++

```
class TRibbon : public TCustomRibbon;
```

File

Ribbon

Description

TRibbon is a TCustomRibbon (see page 3358) that implements the ribbon control. Use TRibbon to create a user interface that looks and performs like the Microsoft Fluent User Interface.

Instead of the tradional user interface where the application has a top level menu, there is no top level menu when using TRibbon. Instead there is an Application Menu (see page 3397), represented by a large round button in the top left corner. Commands are displayed in groups in tabs. These groups are more like toolbars and can display a variety of user interface elements.

See Also

Creating an Application that Uses Ribbon Controls

Adding Commands to the Ribbon

TCustomRibbon (see page 3358)

62.1.16.1 TRibbon Members

The following tables list the members exposed by TRibbon.

Protected Events

	Name	Description
	OnLeftTabScrollButtonClick (see page 3400)	This is OnLeftTabScrollButtonClick, a member of class TCustomRibbon.
	OnRightTabScrollButtonClick (see page 3401)	This is OnRightTabScrollButtonClick, a member of class TCustomRibbon.

Protected Methods

	Name	Description
	CMBidimodechanged (see page 3371)	This is CMBidimodechanged, a member of class TCustomRibbon.
	CMDesignHitTest (see page 3371)	This is CMDesignHitTest, a member of class TCustomRibbon.
	CMFontchanged (see page 3371)	This is CMFontchanged, a member of class TCustomRibbon.
	CMMouseenter (see page 3371)	This is CMMouseenter, a member of class TCustomRibbon.
	CMMouseleave (see page 3372)	This is CMMouseleave, a member of class TCustomRibbon.
	CMTextchanged (see page 3372)	This is CMTextchanged, a member of class TCustomRibbon.
	CMThemeChanged (see page 3372)	This is CMThemeChanged, a member of class TCustomRibbon.
V	ContextMenuPopupHandler (see page 3372)	This is ContextMenuPopupHandler, a member of class TCustomRibbon.

	CreateContextMenu (see page 3373)	This is CreateContextMenu, a member of class TCustomRibbon.
	CustomFrameWndProc (see page 3373)	This is CustomFrameWndProc, a member of class TCustomRibbon.
	DblClick (see page 3373)	OnDblClick event dispatcher.
	DefineProperties (see page 3374)	Provides methods to read and write the IsControl property to a stream such as a form file.
	DisplayKeyTips (see page 3374)	This is DisplayKeyTips, a member of class TCustomRibbon.
	DoAddToQuickAccessToolbar (see page 3375)	This is DoAddToQuickAccessToolbar, a member of class TCustomRibbon.
	DoChangeQuickAccessToolbarPosition (see page 3375)	This is DoChangeQuickAccessToolbarPosition, a member of class TCustomRibbon.
	DoContextPopup (see page 3375)	OnContextPopup event dispatcher.
	DoCustomizeQuickAccessToolbar (see page 3375)	This is DoCustomizeQuickAccessToolbar, a member of class TCustomRibbon.
	DoGroupStyleChanged (see page 3376)	This is DoGroupStyleChanged, a member of class TCustomRibbon.
	DoHelpButtonClick (see page 3376)	This is DoHelpButtonClick, a member of class TCustomRibbon.
	DoMinimizeRibbon (see page 3376)	This is DoMinimizeRibbon, a member of class TCustomRibbon.
	DoMouseWheelDown (see page 3376)	OnMouseWheelDown event dispatcher.
	DoMouseWheelUp (see page 3377)	OnMouseWheelUp event dispatcher.
	DoRecentItemClick (see page 3377)	This is DoRecentItemClick, a member of class TCustomRibbon.
	DoScrollTabsLeft (see page 3378)	This is DoScrollTabsLeft, a member of class TCustomRibbon.
	DoScrollTabsRight (see page 3378)	This is DoScrollTabsRight, a member of class TCustomRibbon.
	DoTabChange (see page 3378)	This is DoTabChange, a member of class TCustomRibbon.
	DrawCaption (see page 3378)	This is DrawCaption, a member of class TCustomRibbon.
	DrawQuickAccessToolbar (see page 3379)	This is DrawQuickAccessToolbar, a member of class TCustomRibbon.
	DrawSystemButtons (see page 3379)	This is DrawSystemButtons, a member of class TCustomRibbon.
	DrawTab (see page 3379)	This is DrawTab, a member of class TCustomRibbon.
	GetContextMenuOptions (see page 3379)	This is GetContextMenuOptions, a member of class TCustomRibbon.
	GetHelpButtonRect (see page 3380)	This is GetHelpButtonRect, a member of class TCustomRibbon.
	GetRibbonHeight (see page 3380)	This is GetRibbonHeight, a member of class TCustomRibbon.
	GetTabAt (see page 3381)	This is GetTabAt, a member of class TCustomRibbon.
	GetTabRect (see page 3381)	This is GetTabRect, a member of class TCustomRibbon.
	GetTabWidth (see page 3381)	This is GetTabWidth, a member of class TCustomRibbon.
	HasApplicationMenu (see page 3381)	This is HasApplicationMenu, a member of class TCustomRibbon.
	HasQuickAccessToolbar (see page 3382)	This is HasQuickAccessToolbar, a member of class TCustomRibbon.
	Idle (see page 3382)	This is Idle, a member of class TCustomRibbon.
	KeyDown (see page 3383)	Respond to key press events.
	Loaded (see page 3384)	Initializes the control after it is loaded from a stream.
	MouseDown (see page 3384)	OnMouseDown event dispatcher.
	MouseInSystemMenu (see page 3385)	This is MouseInSystemMenu, a member of class TCustomRibbon.
	MouseMove (see page 3385)	Respond to mouse moving over control area..
	MouseUp (see page 3385)	OnMouseUp event dispatcher.
	Notification (see page 3386)	Responds to notifications that components are being created or destroyed.
	Paint (see page 3387)	Renders the image of a custom control.
	RebuildRibbonMetrics (see page 3387)	This is RebuildRibbonMetrics, a member of class TCustomRibbon.
	Resize (see page 3388)	Respond to control resize.
	SetParent (see page 3389)	Sets the parent of the control.
	UMCheckResize (see page 3391)	This is UMCheckResize, a member of class TCustomRibbon.
	UMDisplayKeyTips (see page 3391)	This is UMDisplayKeyTips, a member of class TCustomRibbon.
	UpdateQATPosition (see page 3391)	This is UpdateQATPosition, a member of class TCustomRibbon.
	UpdateSystemMenuInfo (see page 3391)	This is UpdateSystemMenuInfo, a member of class TCustomRibbon.
	ValidTab (see page 3392)	This is ValidTab, a member of class TCustomRibbon.
	WMEraseBkgnd (see page 3392)	This is WMEraseBkgnd, a member of class TCustomRibbon.
	WMGetDlgCode (see page 3392)	This is WMGetDlgCode, a member of class TCustomRibbon.
	WMKillFocus (see page 3392)	This is WMKillFocus, a member of class TCustomRibbon.
	WMSettingChange (see page 3393)	This is WMSettingChange, a member of class TCustomRibbon.
	WMWindowPosChanged (see page 3393)	This is WMWindowPosChanged, a member of class TCustomRibbon.

Protected Properties

	Name	Description
	AllowKeyboardNavigation (see page 3395)	This is AllowKeyboardNavigation, a member of class TCustomRibbon.
	HideRibbon (see page 3396)	This is HideRibbon, a member of class TCustomRibbon.
	HotTabIndex (see page 3396)	This is HotTabIndex, a member of class TCustomRibbon.
	KeyTipsVisible (see page 3397)	This is KeyTipsVisible, a member of class TCustomRibbon.
R	Menu (see page 3397)	This is Menu, a member of class TCustomRibbon.
R	ScaleFactor (see page 3398)	This is ScaleFactor, a member of class TCustomRibbon.

Public Constants

	Name	Description
	cRibbonFirstTabOffSet (see page 3364)	This is cRibbonFirstTabOffSet, a member of class TCustomRibbon.
	cRibbonGroupCaptionHeight (see page 3364)	This is cRibbonGroupCaptionHeight, a member of class TCustomRibbon.
	cRibbonGroupHeight (see page 3364)	This is cRibbonGroupHeight, a member of class TCustomRibbon.
	cRibbonHeight (see page 3364)	This is cRibbonHeight, a member of class TCustomRibbon.
	cRibbonHideHeight (see page 3365)	This is cRibbonHideHeight, a member of class TCustomRibbon.
	cRibbonHideWidth (see page 3365)	This is cRibbonHideWidth, a member of class TCustomRibbon.
	cRibbonMinimizedHeight (see page 3365)	This is cRibbonMinimizedHeight, a member of class TCustomRibbon.
	cRibbonMinimumCaptionWidth (see page 3365)	This is cRibbonMinimumCaptionWidth, a member of class TCustomRibbon.
	cRibbonPageHeight (see page 3366)	This is cRibbonPageHeight, a member of class TCustomRibbon.
	cRibbonQATHeight (see page 3366)	This is cRibbonQATHeight, a member of class TCustomRibbon.
	cRibbonQuickAccessToolbarLeft (see page 3366)	This is cRibbonQuickAccessToolbarLeft, a member of class TCustomRibbon.
	cRibbonQuickAccessToolbarTop (see page 3366)	This is cRibbonQuickAccessToolbarTop, a member of class TCustomRibbon.
	cRibbonTabHeight (see page 3366)	This is cRibbonTabHeight, a member of class TCustomRibbon.
	cRibbonTabScrollButtonWidth (see page 3367)	This is cRibbonTabScrollButtonWidth, a member of class TCustomRibbon.
	cRibbonTabSpacing (see page 3367)	This is cRibbonTabSpacing, a member of class TCustomRibbon.
	cRibbonUnthemedCaptionHeight (see page 3367)	This is cRibbonUnthemedCaptionHeight, a member of class TCustomRibbon.
	UM_CHECKSIZE (see page 3367)	This is UM_CHECKSIZE, a member of class TCustomRibbon.
	UM_DISPLAYKEYTIPS (see page 3368)	This is UM_DISPLAYKEYTIPS, a member of class TCustomRibbon.

Public Methods

	Name	Description
	AddRecentItem (see page 3370)	Add new entry into Recent Items list.
	AddTab (see page 3370)	This is AddTab, a member of class TCustomRibbon.
	BeginUpdate (see page 3370)	This is BeginUpdate, a member of class TCustomRibbon.
	CheckRibbonSize (see page 3370)	This is CheckRibbonSize, a member of class TCustomRibbon.
	ClearRecentItems (see page 3371)	This is ClearRecentItems, a member of class TCustomRibbon.
V	Create (see page 3362)	Creates an instance of TCustomControl.
	CustomFrameSupported (see page 3373)	This is CustomFrameSupported, a member of class TCustomRibbon.
	DeleteTab (see page 3374)	Delete tab at specified Index.
V	Destroy (see page 3363)	Destroys an instance of TCustomControl.
	DoMouseWheelEx (see page 3377)	This is DoMouseWheelEx, a member of class TCustomRibbon.
	EndUpdate (see page 3379)	This is EndUpdate, a member of class TCustomRibbon.
V	GetIcon (see page 3380)	Return Icon used by Application Menu (see page 3397) Button.
	GetRibbonMetric (see page 3380)	This is GetRibbonMetric, a member of class TCustomRibbon.
	InvalidateHeaderArea (see page 3382)	Invalidate header section of Ribbon (see page 3347).
	InvalidateScrollButtons (see page 3382)	This is InvalidateScrollButtons, a member of class TCustomRibbon.
	InvalidateSystemMenuArea (see page 3383)	This is InvalidateSystemMenuArea, a member of class TCustomRibbon.
	InvalidateTabArea (see page 3383)	Invalidate area that displays tabs.
	MoveGroup (see page 3386)	Move group to new page.
	RemoveAllTabs (see page 3387)	Remove all Tabs (see page 3399) from Ribbon (see page 3347).
	RemoveRecentItem (see page 3388)	This is RemoveRecentItem, a member of class TCustomRibbon.
	RemoveTab (see page 3388)	Remove tab with specified caption.
V	SetBounds (see page 3389)	Sets the windowed control's boundary properties all at once.

	TabHandledKeyTips (see page 3390)	Validate **KeyTip** input when **KeyTip**s State is **ktsTabs**.
	TabHideKeyTips (see page 3390)	Hide K**eyTip**s.
	TabShowKeyTips (see page 3390)	Display **KeyTip**s for **ktsTabs** State.

Public Nested Types

Name	Description
TTabChangeEvent (see page 3400)	This is nested type Ribbon.TCustomRibbon.TTabChangeEvent.

Public Properties

	Name	Description
	Active (see page 3394)	Indicates if form that ribbon is placed on is active form.
	ActivePage (see page 3394)	Page currently displayed by ribbon control.
	ColorMap (see page 3395)	This is ColorMap, a member of class TCustomRibbon.
	Minimized (see page 3397)	Show Ribbon (see page 3347) in minimized state where tabs can be selected but are not visible.

Published Events

	Name	Description
	OnHelpButtonClick (see page 3498)	This is OnHelpButtonClick, a member of class TRibbon.
	OnRecentItemClick (see page 3498)	Recent item clicked.
	OnTabChange (see page 3498)	This is OnTabChange, a member of class TRibbon.

Published Properties

	Name	Description
	ActionManager (see page 3484)	This is ActionManager, a member of class TRibbon.
	Align (see page 3484)	Determines how the control aligns within its container (parent control).
	Anchors (see page 3488)	Specifies how the control is anchored to its parent.
	ApplicationMenu (see page 3488)	Change appearance of ribbon's associated TRibbonApplicationMenuBar control.
	BiDiMode (see page 3488)	Specifies the bi-directional mode for the control.
	Caption (see page 3489)	Specifies a text string that identifies the control to the user.
	DocumentName (see page 3492)	Text in Ribbon caption.
	Enabled (see page 3492)	Controls whether the control responds to mouse, keyboard, and timer events.
	Font (see page 3492)	Controls the attributes of text written on or in the control.
	Height (see page 3493)	Specifies the vertical size of the control in pixels.
	HideTabs (see page 3495)	Shows or hides Tab portion of Ribbon.
	ParentBiDiMode (see page 3495)	Specifies whether the control uses its parent's BiDiMode.
	ParentFont (see page 3495)	Determines where a control looks for its font information.
	QuickAccessToolbar (see page 3496)	Properties associated with TQuickAccessToolbar.
	ScreenTips (see page 3496)	This is ScreenTips, a member of class TRibbon.
	ShowHelpButton (see page 3496)	Determines whether ribbon displays Help Button.
	Style (see page 3497)	This is Style, a member of class TRibbon.
	TabIndex (see page 3497)	Change the active page of ribbon.
	Tabs (see page 3497)	Collection of TRibbonTabs visible on ribbon.
	UseCustomFrame (see page 3497)	Indicates whether to use a custom frame for ribbon controls.

62.1.16.2 TRibbon Properties

The properties of the TRibbon class are listed here.

Published Properties

	Name	Description
	ActionManager (see page 3484)	This is ActionManager, a member of class TRibbon.
	Align (see page 3484)	Determines how the control aligns within its container (parent control).
	Anchors (see page 3488)	Specifies how the control is anchored to its parent.

	ApplicationMenu (see page 3488)	Change appearance of ribbon's associated TRibbonApplicationMenuBar control.
	BiDiMode (see page 3488)	Specifies the bi-directional mode for the control.
	Caption (see page 3489)	Specifies a text string that identifies the control to the user.
	DocumentName (see page 3492)	Text in Ribbon caption.
	Enabled (see page 3492)	Controls whether the control responds to mouse, keyboard, and timer events.
	Font (see page 3492)	Controls the attributes of text written on or in the control.
	Height (see page 3493)	Specifies the vertical size of the control in pixels.
	HideTabs (see page 3495)	Shows or hides Tab portion of Ribbon.
	ParentBiDiMode (see page 3495)	Specifies whether the control uses its parent's BiDiMode.
	ParentFont (see page 3495)	Determines where a control looks for its font information.
	QuickAccessToolbar (see page 3496)	Properties associated with TQuickAccessToolbar.
	ScreenTips (see page 3496)	This is ScreenTips, a member of class TRibbon.
	ShowHelpButton (see page 3496)	Determines whether ribbon displays Help Button.
	Style (see page 3497)	This is Style, a member of class TRibbon.
	TabIndex (see page 3497)	Change the active page of ribbon.
	Tabs (see page 3497)	Collection of TRibbonTabs visible on ribbon.
	UseCustomFrame (see page 3497)	Indicates whether to use a custom frame for ribbon controls.

62.1.16.2.1 TRibbon.ActionManager Property

Pascal

```
property ActionManager: TCustomActionManager;
```

C++

```
__property TCustomActionManager ActionManager;
```

Description

This is ActionManager, a member of class TRibbon.

62.1.16.2.2 TRibbon.Align Property

Determines how the control aligns within its container (parent control).

Pascal

```
property Align;
```

C++

```
__property Align;
```

Description

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes. When the parent is resized, an aligned control also resizes so that it continues to span the top, bottom, left, or right edge of the parent.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel.

Tip: If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls

into their desired positions.

Note: To cause a control to maintain a specified relationship with an edge of its parent, but not necessarily lie along one edge of the parent, use the Anchors property instead.

C++ Examples:

```
/*
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
*/

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  TSplitter* ps = new TSplitter(Form1); // These objects will all be cleaned up by the owner
(Form1).
  TFileListBox* pflb = new TFileListBox(Form1); // The parent (Form1) will clean up these
objects.
  TDirectoryListBox* pdlb = new TDirectoryListBox(Form1);

  // Line up the directory list box on the left of the form.
  pdlb->Parent = Form1;
  pdlb->Align = alLeft;
  pdlb->Width = Form1->ClientWidth/3;

  // Now use the splitter to divide the directory pane from the file pane.
  ps->Parent = Form1;
  ps->Left = pdlb->Left + pdlb->Width + 1; // Move to right of directory list.
  ps->Align = pdlb->Align; // Give same alignment as directory list.

  // Each pane must be at least one quarter of the form's width.
  ps->MinSize = Form1->ClientWidth/4;

  // Finally, create the last pane - a file list box.
  pflb->Parent = Form1;
  pflb->Align = alClient;
  pdlb->FileList = pflb; // Link the file list box to the directory list box.
}
/*
Dynamically create a TProgressBar control and align it to
the bottom of the form.
*/

#include <ComCtrls.hpp>
#include <Controls.hpp>

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TProgressBar *ProgressBar = new TProgressBar(this);  // The owner will clean this up.
  ProgressBar->Parent = this;
  ProgressBar->Align = alBottom;
  TButton *Button1 = new TButton(this);  // The owner will clean this up.
  Button1->Parent = this;
}
/*
This example dynamically creates a Page Control, then a
series of Tab Sheets on the Page Control.
*/
```

```
#include <Comctrls.hpp>
TPageControl* ppc;
const int MAXTABS = 3;
TTabSheet* pts[MAXTABS];
const char * ppcTabTitles[] = { "ShortString", "Orders", "Items", "Parts" };
int iTabTitles = sizeof(ppcTabTitles)/sizeof(ppcTabTitles[0]);
const TColor colorPalette[12] = {
  clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia,
  clTeal, clNavy, clMaroon, clLime, clOlive, clPurple};

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    ppc = new TPageControl(this); // The owner (this) will clean this up.
    ppc->Parent = this;
    ppc->Align = alClient;
    for (int i=0;i<iTabTitles;i++)
  {
        pts[i] = new TTabSheet(this); // The owner (this) will clean this up.
        pts[i]->PageControl = ppc;
        pts[i]->Name = AnsiString("pts") + ppcTabTitles[i];
        pts[i]->Caption = ppcTabTitles[i];
        pts[i]->Brush->Color = colorPalette[i];
  }
}
```

Delphi Examples:

```
{
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  Split: TSplitter;
  Files: TFileListBox;
  Dirs: TDirectoryListBox;
begin
  { first add a directory list box to the form }
  Dirs := TDirectoryListBox.Create(Form1);
  { line it up on the left of the form }
  Dirs.Parent := Form1;
  Dirs.Align := alLeft;
  Dirs.Width := Form1.ClientWidth div 3;
{ now add the splitter to divide the directory pane from the file pane }
  Split := TSplitter.Create(Form1);
  Split.Parent := Form1;
  { make sure the splitter is to the right of the directory list box! }
  Split.Left := Dirs.Left + Dirs.Width + 1;
  Split.Align := Dirs.Align; { give it the same alignment as the directory }
  { each pane must be at least one quarter of the form's width }
  Split.MinSize := Form1.ClientWidth div 4;
  { Finally, create the last pane - a file list box }
  Files := TFileListBox.Create(Form1);
  Files.Parent := Form1;
  Files.Align := alClient;
  Dirs.FileList := Files; { link the file list box to the directory list box }
end;
{
Dynamically create a TProgressBar control and align it to
the bottom of the form.
}
```

```
var
  I: Integer;

procedure TForm1.Button1Click(Sender: TObject);
var
  ProgressBar: TProgressBar;
begin
  ProgressBar := TProgressBar.Create(Self);
  with ProgressBar do
  begin
    Parent := Self;
    Align := alBottom;
  end;
end;
{
This example dynamically creates a Page Control, then a
series of Tab Sheets on the Page Control.
}
var
  PageControl1: TPageControl;
  pts: array[0..3] of TTabSheet;
const
  TabTitles: array[0..3] of ShortString = ('Customer', 'Orders', 'Items', 'Parts' );

procedure TForm1.FormCreate(Sender: TObject);
const
  colorpalette: Array[0..11] of TColor = (
    clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia, clTeal,
    clNavy, clMaroon, clLime, clOlive, clPurple);
var
  i: Integer;
begin
  PageControl1 := TPageControl.Create(Self);
  PageControl1.Parent := Self;
  PageControl1.Align := alClient;
  for i := Low(TabTitles) to High(TabTitles) do
  begin
    pts[i]:= TTabSheet.Create(PageControl1);
    with pts[i] do
    begin
      PageControl := PageControl1;
      Name := 'ts' + TabTitles[i];
      Caption := TabTitles[i];
      Brush.Color := colorPalette[i];
   end;
  end;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  i: Integer;
begin
  for i := Low(TabTitles) to High(TabTitles) do
  begin
    pts[i].Free;
  end;
  PageControl1.Free;
end;
```

See Also

Parent

Anchors

62.1.16.2.3 TRibbon.Anchors Property

Specifies how the control is anchored to its parent.

Pascal

```
property Anchors;
```

C++

```
__property Anchors;
```

Description

Use Anchors to ensure that a control maintains its current position relative to an edge of its parent, even if the parent is resized. When its parent is resized, the control holds its position relative to the edges to which it is anchored.

If a control is anchored to opposite edges of its parent, the control stretches when its parent is resized. For example, if a control has its Anchors property set to [akLeft, akRight], the control stretches when the width of its parent changes.

Anchors is enforced only when the parent is resized. Thus, for example, if a control is anchored to opposite edges of a form at design time and the form is created in a maximized state, the control is not stretched because the form is not resized after the control is created.

Note: If a control should maintain contact with three edges of its parent (hugging one side of the parent and stretching the length of that side), use the Align property instead. Unlike Anchors, Align allows controls to adjust to changes in the size of other aligned sibling controls as well as changes to the parent's size.

See Also

Parent

Align

62.1.16.2.4 TRibbon.ApplicationMenu Property

Change appearance of ribbon's associated TRibbonApplicationMenuBar control.

Pascal

```
property ApplicationMenu: TApplicationMenu;
```

C++

```
__property TApplicationMenu ApplicationMenu;
```

Description

ApplicationMenu changes the appearance for the ribbon's associated TRibbonApplicationMenuBar control.

Changing the values of the fields in ApplicationMenu changes the characteristics of the Application Menu for the ribbon. The Application Menu on the ribbon is represented by the round button in the top left corner.

See Also

TApplicationMenu (see page 3350)

TRibbonApplicationMenuBar (see page 3942)

62.1.16.2.5 TRibbon.BiDiMode Property

Specifies the bi-directional mode for the control.

Pascal

```
property BiDiMode;
```

C++

```
__property BiDiMode;
```

Description

Use BiDiMode to enable the control to adjust its appearance and behavior automatically when the application runs in a locale that reads from right to left instead of left to right. The bi-directional mode controls the reading order for the text, the placement of the vertical scroll bar, and whether the alignment is changed.

Alignment does not change for controls that are known to contain number, date, time, or currency values. For example, with data aware controls, the alignment does not change for the following field types: ftSmallint, ftInteger, ftWord, ftFloat, ftCurrency, ftBCD, ftDate, ftTime, ftDateTime, ftAutoInc.

See Also

UseRightToLeftAlignment

UseRightToLeftReading

UseRightToLeftScrollBar

Including Bi-directional Functionality in Applications

62.1.16.2.6 TRibbon.Caption Property

Specifies a text string that identifies the control to the user.

Pascal

```
property Caption;
```

C++

```
__property Caption;
```

Description

Use Caption to specify the text string that labels the control.

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

Note: Controls that display text use either the Caption property or the Text property to specify the text value. Which property is used depends on the type of control. In general, Caption is used for text that appears as a window title or label, while Text is used for text that appears as the content of a control.

Note: The Caption property inheritted from TControl is a protected property. Access to the Caption requires casting the class to an inheritted one where Caption is a published or public property.

C++ Examples:

```
/*
The following code keeps the size of the label control
constant, even though the length of the label's caption
changes. As a result, the caption of the label is probably
too long to display in the label when the user clicks the
```

```
button:
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Label1->AutoSize = false;
  Label1->Caption = "This string is too long to be the caption of this label";
}
/*
This example requires a TPageControl populated by several
TabSheets, a TEdit and a TButton. When the button is
clicked, the caption of the active tab changes to that of
the TEdit.  These lines of code could also be written:
PageControl1.ActivePage.Caption := Edit1.Text;
This would make a TTabSheet variable unnecessary since
PageControl1.->ActivePage is already of type TTabSheet.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TTabSheet *sheet = PageControl1->ActivePage;
  sheet->Caption = Edit1->Text;
}
/*
This example creates a ToolBar and adds ToolButtons to it.
Include ComCtlrs in the uses clause.
*/
#include <comctrls.hpp>
#include <memory>

void AddButtons(TToolBar *toolBar, TStringList *captions)
{
  for (int i = 0; i < captions->Count; i++)
  {
    TToolButton *button = new TToolButton(toolBar);  // The owner (toolBar) will clean this up.
    button->Parent = toolBar;
    button->Caption = captions->Strings[i];
    if (button->Caption == "|")
      button->Style = tbsSeparator;
    else
      button->Style = tbsButton;
  }
}

TToolBar *myToolBar;

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  myToolBar = new TToolBar(this); // The owner (this) will clean this up.
  myToolBar->Parent = this;
//  TStringList *captions = new TStringList();
  std::auto_ptr<TStringList> captions(new TStringList());

  captions->Add("New");
  captions->Add("Save");
  captions->Add("|");
  captions->Add("Cut");
  captions->Add("Copy");
  captions->Add("Paste");
  AddButtons(myToolBar, captions.get());
//  delete captions;
  myToolBar->ShowCaptions = True;
  myToolBar->Height = 40;
}
```

Delphi Examples:

```
{
The following code keeps the size of the label control
```

```
constant, even though the length of the label's caption
changes. As a result, the caption of the label is probably
too long to display in the label when the user clicks the
button:
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.AutoSize := False;
  Label1.Caption :=
    'This string is too long to be the caption of this label';
end;
{
This example requires a TPageControl populated by several
TabSheets, a TEdit and a TButton. When the button is
clicked, the caption of the active tab changes to that of
the TEdit.  These lines of code could also be written:
PageControl1.ActivePage.Caption := Edit1.Text;
This would obviate the need for a TTabSheet variable since
PageControl1.->ActivePage is already of type TTabSheet.
}
procedure TForm1.Button1Click(Sender: TObject);
var
  sheet: TTabSheet;
begin
  sheet := PageControl1.ActivePage;
  sheet.Caption := Edit1.Text;
end;
{
This example creates a ToolBar and adds ToolButtons to it.
Include ComCtlrs in the uses clause.
}
procedure AddButtons(ToolBar: TToolBar;
  Const ButtonCaptions: array of String);
var
  I: Integer;
begin
  for I := 0 to High(ButtonCaptions) do
    begin
      with TToolButton.Create(ToolBar) do
      begin
        Parent := ToolBar;
        Caption := ButtonCaptions[I];
        if (ButtonCaptions[I] = '|') then
          Style := tbsSeparator
        else
          Style := tbsButton;
      end;
    end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1 := TToolBar.Create(Self);
  ToolBar1.Parent := Self;
  AddButtons(ToolBar1, ['New', 'Save', '|', 'Cut', 'Copy', 'Paste']);
  ToolBar1.ShowCaptions := True;
  ToolBar1.Height := 40;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  I: Integer;
begin
  for I := ToolBar1.ButtonCount - 1 downto 0 do
    ToolBar1.Buttons[I].Free;
  ToolBar1.Free;
```

```
end;
```

62.1.16.2.7 TRibbon.DocumentName Property

Text in Ribbon caption.

Pascal

```
property DocumentName: string;
```

C++

```
__property UnicodeString DocumentName;
```

Description

DocumentName appears in the Ribbon caption.

62.1.16.2.8 TRibbon.Enabled Property

Controls whether the control responds to mouse, keyboard, and timer events.

Pascal

```
property Enabled;
```

C++

```
__property Enabled;
```

Description

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to False. Some disabled controls appear dimmed (for example: buttons, check boxes, labels), while others (container controls) simply lose their functionality without changing their appearance. If Enabled is set to False, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to True.

Note: This property applies to all TControl descendants.

See Also

Visible

DisableControls

EnableControls

Enabling Mouse

62.1.16.2.9 TRibbon.Font Property

Controls the attributes of text written on or in the control.

Pascal

```
property Font;
```

C++

```
__property Font;
```

Description

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name,

Pitch, Size, or Style of the TFont object.

C++ Examples:

```
/*
To use this example, place a TComboBox and TRichEdit or
TMemo on a form.  During form creation, a list of font names
is loaded into the combo box. When the combo box is clicked,
the memo or Rich Edit control font is set to the
corresponding font name in the combo box.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < Screen->Fonts->Count; i++)
   ComboBox1->Items->Add(Screen->Fonts->Strings[i]);
//  ComboBox1->Items = Screen->Fonts;  // Another way to do it.
}

void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
  RichEdit1->Font->Name = ComboBox1->Text;
}
```

Delphi Examples:

```
{
To use this example, place a TComboBox and TRichEdit or
TMemo on a form.  During form creation, a list of font names
is loaded into the combo box. When the combo box is clicked,
the memo or Rich Edit control font is set to the
corresponding font name in the combo box.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  i : Integer;
begin
  for i := 0 to Screen.Fonts.Count - 1 do
   ComboBox1.Items.Add(Screen.Fonts.Strings[i]);
//  ComboBox1.Items := Screen.Fonts;  // Another way to do it.
  RichEdit1.Lines.Text := 'How is it to be you?';
end;

procedure TForm1.ComboBox1Click(Sender: TObject);
begin
  RichEdit1.Font.Name := ComboBox1.Text;
end;
```

See Also

DesktopFont

ParentFont

Displaying and Editing Text in a Memo Control

62.1.16.2.10 TRibbon.Height Property

Specifies the vertical size of the control in pixels.

Pascal

```
property Height;
```

C++

```
__property Height;
```

Description

Use the Height property to read or change the height of the control.

C++ Examples:

```
/*
This example needs only a blank form.  Three Rich Edit
controls are placed vertically on the form, each contain a
line of text, and the second form has HideSelection set to
false.  Try tabbing between the three Rich Edit controls and
observe the effect on the selected text.
*/
#include <ComCtrls.hpp>

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int I = 0; I <= 2; I++)
  {
    TRichEdit *pRich = new TRichEdit(this);
    pRich->Parent = this;
    pRich->Top = pRich->Height * I;
    pRich->Text = "This text will take the focus.";
    pRich->SelStart = 0;
    pRich->SelLength = pRich->Text.Length();
    if (I == 1)
      pRich->HideSelection = false;
  }
}
```

Delphi Examples:

```
{
This example needs only a blank form. Three Rich Edit
controls are placed vertically on the form, each contain a
line of text, and the second form has HideSelection set to
false.  Try tabbing between the three Rich Edit controls and
observe the effect on the selected text.
}
uses ComCtrls;

procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
  for I := 0 to 2 do
    with TRichEdit.Create(Self) do
    begin
      Parent := Self;
      Top := Height * I;
      Text := 'This text will take the focus.';
      SelStart := 0;
      SelLength := Length(Text);
      if (I = 1) then
        HideSelection := False;
    end;
end;
```

See Also

ClientHeight

Width

Using TDBListBox and TDBComboBox

62.1.16.2.11 TRibbon.HideTabs Property

Shows or hides Tab portion of Ribbon.

Pascal

```
property HideTabs: Boolean;
```

C++

```
__property Boolean HideTabs;
```

Description

HideTabs shows or hides the Tab portion of the Ribbon. When HideTabs is True, the user cannot access any of the Tabs on the Ribbon.

62.1.16.2.12 TRibbon.ParentBiDiMode Property

Specifies whether the control uses its parent's BiDiMode.

Pascal

```
property ParentBiDiMode;
```

C++

```
__property ParentBiDiMode;
```

Description

Use ParentBiDiMode to get or set whether the control uses its parent's BiDiMode. When ParentBiDiMode is true, the control's BiDiMode property is always equal to that of its parent. If ParentBiDiMode is false, the control uses its own BiDiMode property.

Set ParentBiDiMode to true for all controls in a form in order to ensure that the entire user interface adjusts to Middle Eastern locales in a uniform manner.

When the value of the control's BiDiMode property changes, ParentBiDiMode becomes false automatically.

See Also

Parent

BiDiMode

62.1.16.2.13 TRibbon.ParentFont Property

Determines where a control looks for its font information.

Pascal

```
property ParentFont;
```

C++

```
__property ParentFont;
```

Description

To have a control use the same font as its parent control, set ParentFont to true. If ParentFont is false, the control uses its own Font property. Many controls default ParentFont to true so that all the controls in a form or other container present a uniform appearance. When the value of a control's Font property changes, ParentFont becomes false automatically.

When ParentFont is true for a form, the form uses the default font.

See Also

Font

Parent

ParentColor

ParentShowHint

62.1.16.2.14 TRibbon.QuickAccessToolbar Property

Properties associated with TQuickAccessToolbar.

Pascal

```
property QuickAccessToolbar: TQuickAccessToolbar;
```

C++

```
__property TQuickAccessToolbar QuickAccessToolbar;
```

Description

QuickAccessToolbar has the properties associated with the TQuickAccessToolbar on the ribbon. Use these properties to change the appearance of the Quick Access Toolbar as displayed on the ribbon.

See Also

TQuickAccessToolbar (see page 3469)

62.1.16.2.15 TRibbon.ScreenTips Property

Pascal

```
property ScreenTips: TCustomScreenTipsManager;
```

C++

```
__property TCustomScreenTipsManager ScreenTips;
```

Description

This is ScreenTips, a member of class TRibbon.

62.1.16.2.16 TRibbon.ShowHelpButton Property

Determines whether ribbon displays Help Button.

Pascal

```
property ShowHelpButton: Boolean;
```

C++

```
__property Boolean ShowHelpButton;
```

Description

ShowHelpButton indicates whether or not the ribbon displays a Help Button in the top right hand corner.

See Also

THelpButton (see page 3459)

62.1.16.2.17 TRibbon.Style Property

Pascal

```
property Style: TRibbonStyleActionBars;
```

C++

```
__property TRibbonStyleActionBars Style;
```

Description

This is Style, a member of class TRibbon.

62.1.16.2.18 TRibbon.TabIndex Property

Change the active page of ribbon.

Pascal

```
property TabIndex: Integer;
```

C++

```
__property int TabIndex;
```

Description

TabIndex changes the active page of the ribbon. Set the TabIndex property to the desired page index.

See Also

Tabs (see page 3399)

62.1.16.2.19 TRibbon.Tabs Property

Collection of TRibbonTabs visible on ribbon.

Pascal

```
property Tabs: TRibbonTabs;
```

C++

```
__property TRibbonTabs Tabs;
```

Description

Tabs contains a collection of TRibbonTabs visible on the ribbon.

Tabs is a linked list of TRibbonTabs objects. TRibbonTabs is a simple data structure that contains the caption for the tab, the keytip for the tab as well as a reference to the Tab object.

See Also

TRibbonTabs (see page 3560)

TabIndex (see page 3399)

62.1.16.2.20 TRibbon.UseCustomFrame Property

Indicates whether to use a custom frame for ribbon controls.

Pascal

```
property UseCustomFrame: Boolean;
```

C++

```
__property Boolean UseCustomFrame;
```

Description

UseCustomFrame indicates whether or not to use a custom frame for ribbon controls.

62.1.16.3 TRibbon Events

The events of the TRibbon class are listed here.

Published Events

	Name	Description
	OnHelpButtonClick (see page 3498)	This is OnHelpButtonClick, a member of class TRibbon.
	OnRecentItemClick (see page 3498)	Recent item clicked.
	OnTabChange (see page 3498)	This is OnTabChange, a member of class TRibbon.

62.1.16.3.1 TRibbon.OnHelpButtonClick Event

Pascal

```
property OnHelpButtonClick: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnHelpButtonClick;
```

Description

This is OnHelpButtonClick, a member of class TRibbon.

62.1.16.3.2 TRibbon.OnRecentItemClick Event

Recent item clicked.

Pascal

```
property OnRecentItemClick: TRecentItemClickEvent;
```

C++

```
__property TRecentItemClickEvent OnRecentItemClick;
```

Description

OnRecentItemClick occurs when a recent item is clicked.

See Also

TRibbonApplicationMenuBar (see page 3942)

RecentItems (see page 3956)

62.1.16.3.3 TRibbon.OnTabChange Event

Pascal

```
property OnTabChange: TTabChangeEvent;
```

C++

```
__property TTabChangeEvent OnTabChange;
```

Description

This is OnTabChange, a member of class TRibbon.

62.1.17 TRibbonCustomizeDlg Class

Class Hierarchy

Pascal

```
TRibbonCustomizeDlg = class(TCustomizeDlg);
```

C++

```
class TRibbonCustomizeDlg : public TCustomizeDlg;
```

File

Ribbon

Description

This is class Ribbon.TRibbonCustomizeDlg.

62.1.17.1 TRibbonCustomizeDlg Members

The following tables list the members exposed by TRibbonCustomizeDlg.

Protected Methods

	Name	Description
	SetupDlg (see page 3499)	Sets up the customization dialog form.

62.1.17.2 TRibbonCustomizeDlg Methods

The methods of the TRibbonCustomizeDlg class are listed here.

Protected Methods

	Name	Description
	SetupDlg (see page 3499)	Sets up the customization dialog form.

62.1.17.2.1 TRibbonCustomizeDlg.SetupDlg Method

Sets up the customization dialog form.

Pascal

```
procedure SetupDlg; override;
```

C++

```
virtual __fastcall SetupDlg();
```

Description

SetupDlg is a protected method that sets up the customization dialog form.

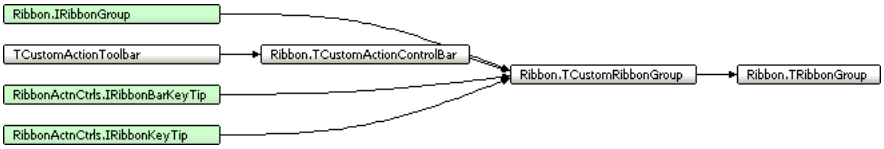
See Also

Show

62.1.18 TRibbonGroup Class

Control used to display the user selectable commands for the ribbon.

Class Hierarchy



Pascal

```
TRibbonGroup = class(TCustomRibbonGroup);
```

C++

```
class TRibbonGroup : public TCustomRibbonGroup;
```

File

Ribbon

Description

TRibbonGroup is a TCustomRibbonGroup (see page 3405) that is used to display the user selectable commands for the ribbon.

When creating a user interface using TRibbon (see page 3480), you add commands to ribbon groups. Ribbon (see page 3347) groups are placed on a ribbon page. You can add new groups to a ribbon by right clicking on the ribbon page and selecting the Add Group component editor item.

The position of the ribbon group is determined by the ribbon page that it was placed on. You cannot freely move the ribbon group around the ribbon page. To change the position of the group within the same page, change the GroupIndex (see page 3504) property.

To move the ribbon group to another ribbon page, use the Move Group command available when right clicking on the ribbon page at design time.

See Also

Creating an Application that Uses Ribbon Controls

Adding Commands to the Ribbon

TRibbon (see page 3480)

TCustomRibbonGroup (see page 3405)

GroupIndex (see page 3422)

62.1.18.1 TRibbonGroup Members

The following tables list the members exposed by TRibbonGroup.

Enumerations

	Name	Description
	TGroupAlign (see page 3408)	Alignment of ribbon group.

Protected Methods

	Name	Description
V	AlignControls (see page 3409)	This is AlignControls, a member of class TCustomRibbonGroup.
V	AutoSizingChanged (see page 3410)	This is AutoSizingChanged, a member of class TCustomRibbonGroup.
	CMControlChange (see page 3357)	This is CMControlChange, a member of class TCustomActionControlBar.
	CMFontchanged (see page 3412)	This is CMFontchanged, a member of class TCustomRibbonGroup.
	CMItemDropPoint (see page 3412)	This is CMItemDropPoint, a member of class TCustomRibbonGroup.
	CMMouseenter (see page 3412)	This is CMMouseenter, a member of class TCustomRibbonGroup.
	CMMouseleave (see page 3412)	This is CMMouseleave, a member of class TCustomRibbonGroup.
	CMTextchanged (see page 3413)	This is CMTextchanged, a member of class TCustomRibbonGroup.
V	CreateControls (see page 3357)	This is CreateControls, a member of class TCustomActionControlBar.
V	CreateControls (see page 3413)	This is CreateControls, a member of class TCustomRibbonGroup.
V	DoDropItem (see page 3414)	This is DoDropItem, a member of class TCustomRibbonGroup.
V	DoEndDrag (see page 3414)	This is DoEndDrag, a member of class TCustomRibbonGroup.
V	DrawShowDialogButton (see page 3414)	This is DrawShowDialogButton, a member of class TCustomRibbonGroup.
	GetCaptionHeight (see page 3414)	This is GetCaptionHeight, a member of class TCustomRibbonGroup.
V	GetControlClass (see page 3357)	This is GetControlClass, a member of class TCustomActionControlBar.
	GetGroupHeight (see page 3415)	This is GetGroupHeight, a member of class TCustomRibbonGroup.
V	HideControls (see page 3416)	This is HideControls, a member of class TCustomRibbonGroup.
V	MouseDown (see page 3417)	This is MouseDown, a member of class TCustomRibbonGroup.
V	MouseMove (see page 3417)	This is MouseMove, a member of class TCustomRibbonGroup.
V	MouseUp (see page 3417)	This is MouseUp, a member of class TCustomRibbonGroup.
V	Notification (see page 3417)	This is Notification, a member of class TCustomRibbonGroup.
V	Paint (see page 3418)	This is Paint, a member of class TCustomRibbonGroup.
V	Resize (see page 3418)	This is Resize, a member of class TCustomRibbonGroup.
V	SetAutoSizing (see page 3418)	This is SetAutoSizing, a member of class TCustomRibbonGroup.
V	SetParent (see page 3419)	This is SetParent, a member of class TCustomRibbonGroup.
V	SetupDropDownBtn (see page 3419)	This is SetupDropDownBtn, a member of class TCustomRibbonGroup.
V	ShowControls (see page 3419)	This is ShowControls, a member of class TCustomRibbonGroup.

Protected Properties

	Name	Description
R	CreatingControls (see page 3358)	This is CreatingControls, a member of class TCustomActionControlBar.

Public Methods

	Name	Description
	AlignGroupControls (see page 3568)	Realign controls on group.
V	AlignGroupControls (see page 3410)	Align controls in Ribbon (see page 3347) Group
V	Assign (see page 3410)	This is Assign, a member of class TCustomRibbonGroup.
	BarHandleKeyTips (see page 3410)	Iterate over any visible **KeyTip (see page 3423)**s and perform validation to determine if one of **KeyTip (see page 3423)**s is partial or full match.
	BarHandleKeyTips (see page 3913)	Check if any commands on Action Bar are partial or full match for **KeyTip**.
	BarHideKeyTips (see page 3411)	Hide any visible KeyTips that commands are displaying.
	BarHideKeyTips (see page 3913)	Hide any visible **KeyTip** windows.
	BarShowKeyTips (see page 3411)	Show K**eyTip**s for commands.

	BarShowKeyTips (see page 3914)	Show **KeyTip**s for any commands that support them.
	Clone (see page 3411)	This is Clone, a member of class TCustomRibbonGroup.
V	Create (see page 3356)	This is Create, a member of class TCustomActionControlBar.
V	Create (see page 3407)	This is Create, a member of class TCustomRibbonGroup.
	CreateControl (see page 3569)	Recreate specific control.
V	CreateControl (see page 3357)	This is CreateControl, a member of class TCustomActionControlBar.
V	CreateControl (see page 3413)	Recreate specific control.
V	Destroy (see page 3408)	This is Destroy, a member of class TCustomRibbonGroup.
V	DockDrop (see page 3413)	This is DockDrop, a member of class TCustomRibbonGroup.
V	DragDrop (see page 3414)	This is DragDrop, a member of class TCustomRibbonGroup.
	GetKeyTip (see page 3415)	Returns current **KeyTip** for command.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	HandleKeyTip (see page 3415)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HideKeyTip (see page 3416)	Hide **KeyTip** for command.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	KeyTipClick (see page 3416)	Handle clicking a **KeyTip**.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
V	SetBounds (see page 3418)	This is SetBounds, a member of class TCustomRibbonGroup.
	SetKeyTip (see page 3418)	Sets **KeyTip** for command.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	ShowKeyTip (see page 3419)	Shows **KeyTip** for command in specified position.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	StyleChanged (see page 3569)	This is StyleChanged, a member of class IRibbonGroup.
V	StyleChanged (see page 3420)	This is StyleChanged, a member of class TCustomRibbonGroup.

Public Nested Types

Name	Description
TRowPosition (see page 3424)	This is nested type Ribbon.TCustomRibbonGroup.TRowPosition.

Public Properties

	Name	Description
R	CanCollapse (see page 3421)	Determine whether group can be displayed collapsed.
R	CollapsedControl (see page 3421)	This is CollapsedControl, a member of class TCustomRibbonGroup.
	GroupVariant (see page 3422)	Current size variant ribbon group is using.
R	GroupWidth (see page 3422)	This is GroupWidth, a member of class TCustomRibbonGroup.
R	HasGallery (see page 3423)	Indicates if group has an in ribbon Gallery.
	Items (see page 3423)	This is Items, a member of class TCustomRibbonGroup.
R	Page (see page 3423)	Reference to Page group is on.
R	RequiredWidth (see page 3424)	Width required to display group.

Published Properties

	Name	Description
	ActionManager (see page 3503)	This is ActionManager, a member of class TRibbonGroup.
	BiDiMode (see page 3503)	This is BiDiMode, a member of class TRibbonGroup.
	Caption (see page 3503)	This is Caption, a member of class TRibbonGroup.
	DialogAction (see page 3504)	Determines whether dialog button displayed in group.
	GroupAlign (see page 3504)	Determines how controls on group are aligned.
	GroupIndex (see page 3504)	Index of group within current tab.
	Height (see page 3505)	This is Height, a member of class TRibbonGroup.
	KeyTip (see page 3505)	Key tip for Dialog button of group.
	ParentBiDiMode (see page 3505)	This is ParentBiDiMode, a member of class TRibbonGroup.
	Rows (see page 3505)	Number of Rows in group has for controls.

62.1.18.2 TRibbonGroup Properties

The properties of the TRibbonGroup class are listed here.

Published Properties

	Name	Description
	ActionManager (see page 3503)	This is ActionManager, a member of class TRibbonGroup.
	BiDiMode (see page 3503)	This is BiDiMode, a member of class TRibbonGroup.
	Caption (see page 3503)	This is Caption, a member of class TRibbonGroup.
	DialogAction (see page 3504)	Determines whether dialog button displayed in group.
	GroupAlign (see page 3504)	Determines how controls on group are aligned.
	GroupIndex (see page 3504)	Index of group within current tab.
	Height (see page 3505)	This is Height, a member of class TRibbonGroup.
	KeyTip (see page 3505)	Key tip for Dialog button of group.
	ParentBiDiMode (see page 3505)	This is ParentBiDiMode, a member of class TRibbonGroup.
	Rows (see page 3505)	Number of Rows in group has for controls.

62.1.18.2.1 TRibbonGroup.ActionManager Property

Pascal

```
property ActionManager;
```

C++

```
__property ActionManager;
```

Description

This is ActionManager, a member of class TRibbonGroup.

62.1.18.2.2 TRibbonGroup.BiDiMode Property

Pascal

```
property BiDiMode;
```

C++

```
__property BiDiMode;
```

Description

This is BiDiMode, a member of class TRibbonGroup.

62.1.18.2.3 TRibbonGroup.Caption Property

Pascal

```
property Caption;
```

C++

```
__property Caption;
```

Description

This is Caption, a member of class TRibbonGroup.

62.1.18.2.4 TRibbonGroup.DialogAction Property

Determines whether dialog button displayed in group.

Pascal

```
property DialogAction: TContainedAction;
```

C++

```
__property TContainedAction DialogAction;
```

Description

DialogAction determines whether the dialog button is displayed in the lower right hand corner of the group.

See Also

KeyTip (see page 3423)

62.1.18.2.5 TRibbonGroup.GroupAlign Property

Determines how controls on group are aligned.

Pascal

```
property GroupAlign: TGroupAlign;
```

C++

```
__property TGroupAlign GroupAlign;
```

Description

GroupAlign contains a TGroupAlign that determines how controls on a group are aligned.

If alignment is **gaHorizontal**, controls are aligned left to right, top to bottom. To make a control start in a new row, set the NewRow property to true.

If alignment is **gaVertical**, controls are aligned top to bottom, left to right. Controls are automatically moved to the next column in the group if three controls are already in the column. You can also force a control to a new column by setting the NewCol property to true.

See Also

TGroupAlign (see page 3408)

62.1.18.2.6 TRibbonGroup.GroupIndex Property

Index of group within current tab.

Pascal

```
property GroupIndex: Integer;
```

C++

```
__property int GroupIndex;
```

Description

GroupIndex is the index of the group within the current tab. Set the GroupIndex property to change the position of the group within the current tab.

62.1.18.2.7 TRibbonGroup.Height Property

Pascal

```
property Height;
```

C++

```
__property Height;
```

Description

This is Height, a member of class TRibbonGroup.

62.1.18.2.8 TRibbonGroup.KeyTip Property

Key tip for Dialog button of group.

Pascal

```
property KeyTip: string;
```

C++

```
__property UnicodeString KeyTip;__property UnicodeString KeyTip;
```

Description

KeyTip is the key tip for the Dialog button of the group. It is only valid when DialogAction has been set.

See Also

DialogAction (see page 3421)

62.1.18.2.9 TRibbonGroup.ParentBiDiMode Property

Pascal

```
property ParentBiDiMode;
```

C++

```
__property ParentBiDiMode;
```

Description

This is ParentBiDiMode, a member of class TRibbonGroup.

62.1.18.2.10 TRibbonGroup.Rows Property

Number of Rows in group has for controls.

Pascal

```
property Rows: Integer;
```

C++

```
__property int Rows;
```

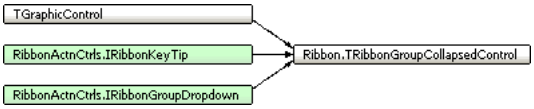
Description

Rows is the number of Rows the group has for displaying controls. A group must have at least 1 row and a maximum of 3.

62.1.19 TRibbonGroupCollapsedControl Class

Control used to represent a collapsed group on a tab.

Class Hierarchy



Pascal

```
TRibbonGroupCollapsedControl = class(TGraphicControl, IRibbonKeyTip, IRibbonGroupDropdown);
```

C++

```
class TRibbonGroupCollapsedControl : public TGraphicControl, public IRibbonKeyTip, public
IRibbonGroupDropdown;
```

File

Ribbon

Description

TRibbonGroupCollapsedControl is a control used to represent a collapsed group on a tab.

This class is used internally by the ribbon group to represent a group in the collapsed state. A group is collapsed when the GroupVariant property of the ribbon group is **rgvCollapsed**. This happens automatically when the ribbon is resized in such a way that the group is no longer completely visible within the current tab.

Note: This class is for internal use only.

See Also

GroupVariant (see page 3422)

62.1.19.1 TRibbonGroupCollapsedControl Members

The following tables list the members exposed by TRibbonGroupCollapsedControl.

Protected Methods

	Name	Description
	AdjustBounds (see page 3509)	This is AdjustBounds, a member of class TRibbonGroupCollapsedControl.
V	Click (see page 3509)	Respond to user click.
	CMFontchanged (see page 3510)	This is CMFontchanged, a member of class TRibbonGroupCollapsedControl.
	CMMouseenter (see page 3510)	This is CMMouseenter, a member of class TRibbonGroupCollapsedControl.
	CMMouseleave (see page 3510)	This is CMMouseleave, a member of class TRibbonGroupCollapsedControl.
	CMTextchanged (see page 3510)	This is CMTextchanged, a member of class TRibbonGroupCollapsedControl.
V	Notification (see page 3512)	Responds to notifications that components are being created or destroyed.
V	Paint (see page 3513)	Render control surface.
	SetupGroupImage (see page 3514)	This is SetupGroupImage, a member of class TRibbonGroupCollapsedControl.

Protected Properties

	Name	Description
	MouseInControl (see page 3518)	This is MouseInControl, a member of class TRibbonGroupCollapsedControl.

Public Methods

	Name	Description
	CloseUp (see page 3509)	This is CloseUp, a member of class TRibbonGroupCollapsedControl.
	Create (see page 3507)	Creates an instance of TGraphicControl.
	Destroy (see page 3508)	Destroys an instance of TGraphicControl.
	GetKeyTip (see page 3510)	Returns current **KeyTip** for command.
	GetParentControl (see page 3511)	This is GetParentControl, a member of class TRibbonGroupCollapsedControl.
	GetTopPosition (see page 3511)	This is GetTopPosition, a member of class TRibbonGroupCollapsedControl.
	HandleKeyTip (see page 3511)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3512)	Hide **KeyTip** for command.
	KeyTipClick (see page 3512)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3513)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3514)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	ActionManager (see page 3515)	This is ActionManager, a member of class TRibbonGroupCollapsedControl.
	OldWidth (see page 3518)	This is OldWidth, a member of class TRibbonGroupCollapsedControl.

Published Properties

	Name	Description
	Caption (see page 3515)	Specifies a text string that identifies the control to the user.
	Group (see page 3518)	Group associated with collapsed button.
	KeyTip (see page 3518)	Current **KeyTip** for command

62.1.19.2 TRibbonGroupCollapsedControl.Create Constructor

Creates an instance of TGraphicControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonGroupCollapsedControl(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TGraphicControl. However, you should never attempt to instantiate a TGraphicControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

AOwner is the component, typically a form, that is responsible for freeing the graphic control. It becomes the value of the Owner property. If you do not explicitly provide an Owner to the constructor (that is, if AOwner is nil (Delphi) or NULL (C++)), your application is responsible for explicitly freeing the graphic control.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Owner

Create

Destroy

62.1.19.3 TRibbonGroupCollapsedControl.Destroy Destructor

Destroys an instance of TGraphicControl.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRibbonGroupCollapsedControl();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the graphic control reference is not nil, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

As the control is destroyed, it:

Frees the TCanvas object in its Canvas property.

Override Destroy to free any memory or resources allocated in the Create method. When declaring a Destroy method in a descendant control type, always add the override directive to the declaration and call the inherited Destroy as the last statement in the redeclared method.

When a control is created, Delphi automatically provides exception handling around the constructor call. The destructor is called if an exception escapes from the constructor. This means that the destructor code must be prepared to clean up a partially constructed instance. Check all data fields for zero before disposing of their contents.

See Also

Create

System::TObject::Free

62.1.19.4 TRibbonGroupCollapsedControl Methods

The methods of the TRibbonGroupCollapsedControl class are listed here.

Protected Methods

	Name	Description
	AdjustBounds (see page 3509)	This is AdjustBounds, a member of class TRibbonGroupCollapsedControl.
V	Click (see page 3509)	Respond to user click.
	CMFontchanged (see page 3510)	This is CMFontchanged, a member of class TRibbonGroupCollapsedControl.
	CMMouseenter (see page 3510)	This is CMMouseenter, a member of class TRibbonGroupCollapsedControl.
	CMMouseleave (see page 3510)	This is CMMouseleave, a member of class TRibbonGroupCollapsedControl.
	CMTextchanged (see page 3510)	This is CMTextchanged, a member of class TRibbonGroupCollapsedControl.
V	Notification (see page 3512)	Responds to notifications that components are being created or destroyed.
V	Paint (see page 3513)	Render control surface.
	SetupGroupImage (see page 3514)	This is SetupGroupImage, a member of class TRibbonGroupCollapsedControl.

Public Methods

	Name	Description
	CloseUp (see page 3509)	This is CloseUp, a member of class TRibbonGroupCollapsedControl.
	GetKeyTip (see page 3510)	Returns current **KeyTip** for command.
	GetParentControl (see page 3511)	This is GetParentControl, a member of class TRibbonGroupCollapsedControl.

	GetTopPosition (see page 3511)	This is GetTopPosition, a member of class TRibbonGroupCollapsedControl.
	HandleKeyTip (see page 3511)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3512)	Hide **KeyTip** for command.
	KeyTipClick (see page 3512)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3513)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3514)	Shows **KeyTip** for command in specified position.

62.1.19.4.1 TRibbonGroupCollapsedControl.AdjustBounds Method

Pascal

```
procedure AdjustBounds; dynamic;
```

C++

```
__fastcall AdjustBounds();
```

Description

This is AdjustBounds, a member of class TRibbonGroupCollapsedControl.

62.1.19.4.2 TRibbonGroupCollapsedControl.Click Method

Respond to user click.

Pascal

```
procedure Click; override;
```

C++

```
virtual __fastcall Click();
```

Description

Click is called automatically when the user left-clicks the control. Component or application code can call Click to simulate a user mouse click. This is often done in menu actions and hotkey handlers.

As implemented in TControl, Click queries whether the OnClick event handler exists and is different from the OnExecute handler for the control's Action. If this is true, the OnClick event handler is called. If this is false, and the Action OnExecute event handler exists, the Action's Execute method is called. Override Click to provide additional behavior.

See Also

Action

OnExecute

Execute

DblClick

OnClick

62.1.19.4.3 TRibbonGroupCollapsedControl.CloseUp Method

Pascal

```
procedure CloseUp;
```

C++

```
__fastcall CloseUp();
```

Description

This is CloseUp, a member of class TRibbonGroupCollapsedControl.

62.1.19.4.4 TRibbonGroupCollapsedControl.CMFontchanged Method

Pascal

```
procedure CMFontchanged(var Message: TMessage); message CM_FONTCHANGED;
```

C++

```
__fastcall CMFontchanged(TMessage Message);
```

Description

This is CMFontchanged, a member of class TRibbonGroupCollapsedControl.

62.1.19.4.5 TRibbonGroupCollapsedControl.CMMouseenter Method

Pascal

```
procedure CMMouseenter(var Message: TMessage); message CM_MOUSEENTER;
```

C++

```
__fastcall CMMouseenter(TMessage Message);
```

Description

This is CMMouseenter, a member of class TRibbonGroupCollapsedControl.

62.1.19.4.6 TRibbonGroupCollapsedControl.CMMouseleave Method

Pascal

```
procedure CMMouseleave(var Message: TMessage); message CM_MOUSELEAVE;
```

C++

```
__fastcall CMMouseleave(TMessage Message);
```

Description

This is CMMouseleave, a member of class TRibbonGroupCollapsedControl.

62.1.19.4.7 TRibbonGroupCollapsedControl.CMTextchanged Method

Pascal

```
procedure CMTextchanged(var Message: TMessage); message CM_TEXTCHANGED;
```

C++

```
__fastcall CMTextchanged(TMessage Message);
```

Description

This is CMTextchanged, a member of class TRibbonGroupCollapsedControl.

62.1.19.4.8 TRibbonGroupCollapsedControl.GetKeyTip Method

Returns current **KeyTip** for command.

Pascal

```
function GetKeyTip: string;
```

C++

```
__fastcall UnicodeString GetKeyTip();
```

Description

GetKeyTip returns the current **KeyTip** for the command

See Also

KeyTip (see page 3922)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

62.1.19.4.9 TRibbonGroupCollapsedControl.GetParentControl Method

Pascal

```
function GetParentControl: TWinControl;
```

C++

```
__fastcall TWinControl * GetParentControl();
```

Description

This is GetParentControl, a member of class TRibbonGroupCollapsedControl.

62.1.19.4.10 TRibbonGroupCollapsedControl.GetTopPosition Method

Pascal

```
function GetTopPosition: Integer;
```

C++

```
__fastcall int GetTopPosition();
```

Description

This is GetTopPosition, a member of class TRibbonGroupCollapsedControl.

62.1.19.4.11 TRibbonGroupCollapsedControl.HandleKeyTip Method

Indicates if command's **KeyTip** matches currently input KeyTip.

Pascal

```
procedure HandleKeyTip(const KeyTip: string; var Handled: TKeyTipHandled);
```

C++

```
__fastcall HandleKeyTip(const UnicodeString KeyTip, TKeyTipHandled Handled);
```

Description

HandleKeyTip indicates whether this command's

KeyTip matches the currently input KeyTip by the user. Full and partial matches are taken into consideration.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

62.1.19.4.12 TRibbonGroupCollapsedControl.HideKeyTip Method

Hide **KeyTip** for command.

Pascal

```
procedure HideKeyTip(const Match: string);
```

C++

```
__fastcall HideKeyTip(const UnicodeString Match);
```

Description

This method hides the **KeyTip** for this command. If the **Match** parameter matches (full or partial) the command's **KeyTip**, the **KeyTip** remains visible.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

62.1.19.4.13 TRibbonGroupCollapsedControl.KeyTipClick Method

Handle clicking a **KeyTip**.

Pascal

```
procedure KeyTipClick;
```

C++

```
__fastcall KeyTipClick();
```

Description

KeyTipClick Handle clicking a **KeyTip**. Some requires need to be handled differently when clicked by a **KeyTip**.

62.1.19.4.14 TRibbonGroupCollapsedControl.Notification Method

Responds to notifications that components are being created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification allows TControl to update its PopupMenu property when the pop-up menu it refers to is destroyed.

Override the Notification method to respond to notifications that other components are about to be destroyed or have just been created. Use the Notification method to update controls that rely on other objects. Remove references to objects that are going away so that the control does not refer to objects that have been destroyed. For example, data-aware objects override the Notification method to update themselves when their data source is removed. Some objects respond to notifications that other objects of a specific type are being created. For example, the session component sets the Session property of new data-aware controls to itself when the AutoSessionName property is true.

By default, components pass along the notification to their owned components, if any.

See Also

PopupMenu

Notification

62.1.19.4.15 TRibbonGroupCollapsedControl.Paint Method

Render control surface.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method is called automatically when a lightweight control needs update its display area. Code in the Paint method uses the Canvas property to render the control surface.

As implemented in TGraphicControl, Paint does nothing. Specific rendering logic is provided by Paint methods in descendant control classes.

See Also

Canvas

62.1.19.4.16 TRibbonGroupCollapsedControl.SetKeyTip Method

Sets **KeyTip** for command.

Pascal

```
procedure SetKeyTip(const Value: string);
```

C++

```
__fastcall SetKeyTip(const UnicodeString Value);
```

Description

This method sets the **KeyTip** for the command.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

ShowKeyTip (see page 3921)

62.1.19.4.17 TRibbonGroupCollapsedControl.SetupGroupImage Method

Pascal

```
procedure SetupGroupImage;
```

C++

```
__fastcall SetupGroupImage();
```

Description

This is SetupGroupImage, a member of class TRibbonGroupCollapsedControl.

62.1.19.4.18 TRibbonGroupCollapsedControl.ShowKeyTip Method

Shows **KeyTip** for command in specified position.

Pascal

```
procedure ShowKeyTip(const Position: TKeyTipPosition);
```

C++

```
__fastcall ShowKeyTip(const TKeyTipPosition Position);
```

Description

This method shows the **KeyTip** for the command in the position specified. Only the Application Button uses a **KeyTip** position.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

62.1.19.5 TRibbonGroupCollapsedControl Properties

The properties of the TRibbonGroupCollapsedControl class are listed here.

Protected Properties

	Name	Description
	MouseInControl (see page 3518)	This is MouseInControl, a member of class TRibbonGroupCollapsedControl.

Public Properties

	Name	Description
	ActionManager (see page 3515)	This is ActionManager, a member of class TRibbonGroupCollapsedControl.
	OldWidth (see page 3518)	This is OldWidth, a member of class TRibbonGroupCollapsedControl.

Published Properties

	Name	Description
	Caption (see page 3515)	Specifies a text string that identifies the control to the user.
	Group (see page 3518)	Group associated with collapsed button.
	KeyTip (see page 3518)	Current **KeyTip** for command

62.1.19.5.1 TRibbonGroupCollapsedControl.ActionManager Property

Pascal

```
property ActionManager: TCustomActionManager;
```

C++

```
__property TCustomActionManager ActionManager;
```

Description

This is ActionManager, a member of class TRibbonGroupCollapsedControl.

62.1.19.5.2 TRibbonGroupCollapsedControl.Caption Property

Specifies a text string that identifies the control to the user.

Pascal

```
property Caption;
```

C++

```
__property Caption;
```

Description

Use Caption to specify the text string that labels the control.

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

Note: Controls that display text use either the Caption property or the Text property to specify the text value. Which property is used depends on the type of control. In general, Caption is used for text that appears as a window title or label, while Text is used for text that appears as the content of a control.

Note: The Caption property inheritted from TControl is a protected property. Access to the Caption requires casting the class to an inheritted one where Caption is a published or public property.

C++ Examples:

```
/*
The following code keeps the size of the label control
constant, even though the length of the label's caption
changes. As a result, the caption of the label is probably
too long to display in the label when the user clicks the
button:
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Label1->AutoSize = false;
```

```
  Label1->Caption = "This string is too long to be the caption of this label";
}
/*
This example requires a TPageControl populated by several
TabSheets, a TEdit and a TButton. When the button is
clicked, the caption of the active tab changes to that of
the TEdit.  These lines of code could also be written:
PageControl1.ActivePage.Caption := Edit1.Text;
This would make a TTabSheet variable unnecessary since
PageControl1.->ActivePage is already of type TTabSheet.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TTabSheet *sheet = PageControl1->ActivePage;
  sheet->Caption = Edit1->Text;
}
/*
This example creates a ToolBar and adds ToolButtons to it.
Include ComCtlrs in the uses clause.
*/
#include <comctrls.hpp>
#include <memory>

void AddButtons(TToolBar *toolBar, TStringList *captions)
{
  for (int i = 0; i < captions->Count; i++)
  {
    TToolButton *button = new TToolButton(toolBar);  // The owner (toolBar) will clean this up.
    button->Parent = toolBar;
    button->Caption = captions->Strings[i];
    if (button->Caption == "|")
      button->Style = tbsSeparator;
    else
      button->Style = tbsButton;
  }
}

TToolBar *myToolBar;

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  myToolBar = new TToolBar(this); // The owner (this) will clean this up.
  myToolBar->Parent = this;
//  TStringList *captions = new TStringList();
  std::auto_ptr<TStringList> captions(new TStringList());

  captions->Add("New");
  captions->Add("Save");
  captions->Add("|");
  captions->Add("Cut");
  captions->Add("Copy");
  captions->Add("Paste");
  AddButtons(myToolBar, captions.get());
//  delete captions;
  myToolBar->ShowCaptions = True;
  myToolBar->Height = 40;
}
```

Delphi Examples:

```
{
The following code keeps the size of the label control
constant, even though the length of the label's caption
changes. As a result, the caption of the label is probably
too long to display in the label when the user clicks the
button:
}
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.AutoSize := False;
  Label1.Caption :=
    'This string is too long to be the caption of this label';
end;
{
This example requires a TPageControl populated by several
TabSheets, a TEdit and a TButton. When the button is
clicked, the caption of the active tab changes to that of
the TEdit.  These lines of code could also be written:
PageControl1.ActivePage.Caption := Edit1.Text;
This would obviate the need for a TTabSheet variable since
PageControl1.->ActivePage is already of type TTabSheet.
}
procedure TForm1.Button1Click(Sender: TObject);
var
  sheet: TTabSheet;
begin
  sheet := PageControl1.ActivePage;
  sheet.Caption := Edit1.Text;
end;
{
This example creates a ToolBar and adds ToolButtons to it.
Include ComCtlrs in the uses clause.
}
procedure AddButtons(ToolBar: TToolBar;
  Const ButtonCaptions: array of String);
var
  I: Integer;
begin
  for I := 0 to High(ButtonCaptions) do
    begin
      with TToolButton.Create(ToolBar) do
      begin
        Parent := ToolBar;
        Caption := ButtonCaptions[I];
        if (ButtonCaptions[I] = '|') then
          Style := tbsSeparator
        else
          Style := tbsButton;
      end;
    end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1 := TToolBar.Create(Self);
  ToolBar1.Parent := Self;
  AddButtons(ToolBar1, ['New', 'Save', '|', 'Cut', 'Copy', 'Paste']);
  ToolBar1.ShowCaptions := True;
  ToolBar1.Height := 40;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  I: Integer;
begin
  for I := ToolBar1.ButtonCount - 1 downto 0 do
    ToolBar1.Buttons[I].Free;
  ToolBar1.Free;
end;
```

62.1.19.5.3 TRibbonGroupCollapsedControl.Group Property

Group associated with collapsed button.

Pascal

```
property Group: TCustomRibbonGroup;
```

C++

```
__property TCustomRibbonGroup Group;
```

Description

Group is the group associated with the collapsed button.

62.1.19.5.4 TRibbonGroupCollapsedControl.KeyTip Property

Current **KeyTip** for command

Pascal

```
property KeyTip: string;
```

C++

```
__property UnicodeString KeyTip;
```

Description

KeyTip is the current **KeyTip** for the command

See Also

GetKeyTip (see page 3919)

SetKeyTip (see page 3921)

62.1.19.5.5 TRibbonGroupCollapsedControl.MouseInControl Property

Pascal

```
property MouseInControl: Boolean;
```

C++

```
__property Boolean MouseInControl;
```

Description

This is MouseInControl, a member of class TRibbonGroupCollapsedControl.

62.1.19.5.6 TRibbonGroupCollapsedControl.OldWidth Property

Pascal

```
property OldWidth: Integer;
```

C++

```
__property int OldWidth;
```

Description

This is OldWidth, a member of class TRibbonGroupCollapsedControl.

62.1.20 TRibbonGroupDockForm Class

Display a collapsed group.

Class Hierarchy

Pascal

```
TRibbonGroupDockForm = class(TXToolDockForm);
```

C++

```
class TRibbonGroupDockForm : public TXToolDockForm;
```

File

Ribbon

Description

TRibbonGroupDockForm is used to display a collapsed group.

When the user clicks on a groups collapsed button, the large group variant of the ribbon group is displayed as a child to an instance of this class.

Note: This class is for internal use only.

62.1.20.1 TRibbonGroupDockForm Members

The following tables list the members exposed by TRibbonGroupDockForm.

Protected Methods

	Name	Description
	CreateParams (see page 3520)	Initializes the window-creation parameter record when the form window is created.
	WMActivate (see page 3520)	This is WMActivate, a member of class TRibbonGroupDockForm.

Public Methods

	Name	Description
	Create (see page 3519)	Creates an instance of TToolDockForm.

Public Properties

	Name	Description
	Control (see page 3521)	Reference to invoking control.
	ParentControl (see page 3521)	This is ParentControl, a member of class TRibbonGroupDockForm.

62.1.20.2 TRibbonGroupDockForm.Create Constructor

Creates an instance of TToolDockForm.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonGroupDockForm(TComponent * AOwner);
```

Description

AOwner is the dock form's owner, which is responsible for freeing the tool dock form.

The constructor initializes the BorderWidth property to 1.

See Also

BorderWidth

Owner

62.1.20.3 TRibbonGroupDockForm Methods

The methods of the TRibbonGroupDockForm class are listed here.

Protected Methods

	Name	Description
	CreateParams (see page 3520)	Initializes the window-creation parameter record when the form window is created.
	WMActivate (see page 3520)	This is WMActivate, a member of class TRibbonGroupDockForm.

62.1.20.3.1 TRibbonGroupDockForm.CreateParams Method

Initializes the window-creation parameter record when the form window is created.

Pascal

```
procedure CreateParams(var Params: TCreateParams); override;
```

C++

```
virtual __fastcall CreateParams(TCreateParams Params);
```

Description

CreateParams calls the inherited CreateParams method, and then adds WS_POPUP to the Style property.

62.1.20.3.2 TRibbonGroupDockForm.WMActivate Method

Pascal

```
procedure WMActivate(var Message: TWMActivate); message WM_ACTIVATE;
```

C++

```
__fastcall WMActivate(TWMActivate Message);
```

Description

This is WMActivate, a member of class TRibbonGroupDockForm.

62.1.20.4 TRibbonGroupDockForm Properties

The properties of the TRibbonGroupDockForm class are listed here.

Public Properties

	Name	Description
	Control (see page 3521)	Reference to invoking control.
	ParentControl (see page 3521)	This is ParentControl, a member of class TRibbonGroupDockForm.

62.1.20.4.1 TRibbonGroupDockForm.Control Property

Reference to invoking control.

Pascal

```
property Control: TControl;
```

C++

```
__property TControl * Control;
```

Description

Control is a reference to the control that invoked the floating group.

62.1.20.4.2 TRibbonGroupDockForm.ParentControl Property

Pascal

```
property ParentControl: TWinControl;
```

C++

```
__property TWinControl * ParentControl;
```

Description

This is ParentControl, a member of class TRibbonGroupDockForm.

62.1.21 TRibbonGroupScrollButton Class

Class Hierarchy

Pascal

```
TRibbonGroupScrollButton = class(TCustomControl);
```

C++

```
class TRibbonGroupScrollButton : public TCustomControl;
```

File

Ribbon

Description

This is class Ribbon.TRibbonGroupScrollButton.

62.1.21.1 TRibbonGroupScrollButton Members

The following tables list the members exposed by TRibbonGroupScrollButton.

Protected Methods

	Name	Description
	CMDesignHitTest (see page 3523)	This is CMDesignHitTest, a member of class TRibbonGroupScrollButton.
	CMMouseenter (see page 3523)	This is CMMouseenter, a member of class TRibbonGroupScrollButton.
	CMMouseleave (see page 3523)	This is CMMouseleave, a member of class TRibbonGroupScrollButton.
V	Paint (see page 3523)	Renders the image of a custom control.

Protected Properties

	Name	Description
	Direction (see page 3524)	This is Direction, a member of class TRibbonGroupScrollButton.
	MouseInControl (see page 3524)	This is MouseInControl, a member of class TRibbonGroupScrollButton.
	RibbonStyle (see page 3524)	This is RibbonStyle, a member of class TRibbonGroupScrollButton.

Public Methods

	Name	Description
V	Create (see page 3522)	Creates an instance of TCustomControl.

62.1.21.2 TRibbonGroupScrollButton.Create Constructor

Creates an instance of TCustomControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonGroupScrollButton(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomControl. However, you should never attempt to instantiate a TCustomControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants. Controls placed on forms at design time are created automatically.

After calling the inherited constructor, Create initializes the control, including the following property settings:

Initializes a TCanvas object for its Canvas property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Create

Destroy

Canvas

Owner

62.1.21.3 TRibbonGroupScrollButton Methods

The methods of the TRibbonGroupScrollButton class are listed here.

Protected Methods

	Name	Description
	CMDesignHitTest (see page 3523)	This is CMDesignHitTest, a member of class TRibbonGroupScrollButton.
	CMMouseenter (see page 3523)	This is CMMouseenter, a member of class TRibbonGroupScrollButton.
	CMMouseleave (see page 3523)	This is CMMouseleave, a member of class TRibbonGroupScrollButton.
	Paint (see page 3523)	Renders the image of a custom control.

62.1.21.3.1 TRibbonGroupScrollButton.CMDesignHitTest Method

Pascal

```
procedure CMDesignHitTest(var Message: TCMDesignHitTest); message CM_DESIGNHITTEST;
```

C++

```
__fastcall CMDesignHitTest(TCMDesignHitTest Message);
```

Description

This is CMDesignHitTest, a member of class TRibbonGroupScrollButton.

62.1.21.3.2 TRibbonGroupScrollButton.CMMouseenter Method

Pascal

```
procedure CMMouseenter(var Message: TMessage); message CM_MOUSEENTER;
```

C++

```
__fastcall CMMouseenter(TMessage Message);
```

Description

This is CMMouseenter, a member of class TRibbonGroupScrollButton.

62.1.21.3.3 TRibbonGroupScrollButton.CMMouseleave Method

Pascal

```
procedure CMMouseleave(var Message: TMessage); message CM_MOUSELEAVE;
```

C++

```
__fastcall CMMouseleave(TMessage Message);
```

Description

This is CMMouseleave, a member of class TRibbonGroupScrollButton.

62.1.21.3.4 TRibbonGroupScrollButton.Paint Method

Renders the image of a custom control.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method for TCustomControl does nothing other than provide the interface for a method that responds to WM_PAINT

messages. Paint is called by the PaintWindow method, after that method has supplied the Canvas with the handle to a device context. When creating a custom control, always override Paint to draw the image of the control.

Tip: To determine which portions of the control's canvas need to be repainted when Paint is called, use the ClipRect property of the canvas.

See Also

Canvas

PaintWindow

GetDeviceContext

ClipRect

62.1.21.4 TRibbonGroupScrollButton Properties

The properties of the TRibbonGroupScrollButton class are listed here.

Protected Properties

	Name	Description
	Direction (see page 3524)	This is Direction, a member of class TRibbonGroupScrollButton.
	MouseInControl (see page 3524)	This is MouseInControl, a member of class TRibbonGroupScrollButton.
	RibbonStyle (see page 3524)	This is RibbonStyle, a member of class TRibbonGroupScrollButton.

62.1.21.4.1 TRibbonGroupScrollButton.Direction Property

Pascal

```
property Direction: TScrollDirection;
```

C++

```
__property TScrollDirection Direction;
```

Description

This is Direction, a member of class TRibbonGroupScrollButton.

62.1.21.4.2 TRibbonGroupScrollButton.MouseInControl Property

Pascal

```
property MouseInControl: Boolean;
```

C++

```
__property Boolean MouseInControl;
```

Description

This is MouseInControl, a member of class TRibbonGroupScrollButton.

62.1.21.4.3 TRibbonGroupScrollButton.RibbonStyle Property

Pascal

```
property RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__property TRibbonStyleActionBars RibbonStyle;
```

Description

This is RibbonStyle, a member of class TRibbonGroupScrollButton.

62.1.22 TRibbonLunaColorMap Class

Color map class for Luna.

Class Hierarchy

Pascal

```
TRibbonLunaColorMap = class(TCustomRibbonColorMap);
```

C++

```
class TRibbonLunaColorMap : public TCustomRibbonColorMap;
```

File

Ribbon

Description

TRibbonLunaColorMap is the color map class for the Luna ribbon style.

See Also

TRibbonObsidianColorMap (see page 3530)

TRibbonSilverColorMap (see page 3549)

62.1.22.1 TRibbonLunaColorMap Members

The following tables list the members exposed by TRibbonLunaColorMap.

Public Methods

	Name	Description
	UpdateColors (see page 3526)	Updates an action component to reflect the current color property.

Published Properties

	Name	Description
	ActiveTabFontColor (see page 3527)	This is ActiveTabFontColor, a member of class TRibbonLunaColorMap.
	BevelMinimizedInner (see page 3527)	This is BevelMinimizedInner, a member of class TRibbonLunaColorMap.
	BevelMinimizedOuter (see page 3527)	This is BevelMinimizedOuter, a member of class TRibbonLunaColorMap.
	BtnSelectedFont (see page 3527)	Changes the look and feel of your applications.
	CaptionFontColor (see page 3528)	This is CaptionFontColor, a member of class TRibbonLunaColorMap.
	ControlColor (see page 3528)	This is ControlColor, a member of class TRibbonLunaColorMap.
	ControlFrameColor (see page 3528)	This is ControlFrameColor, a member of class TRibbonLunaColorMap.
	ControlInactiveColor (see page 3528)	This is ControlInactiveColor, a member of class TRibbonLunaColorMap.
	DisabledFontColor (see page 3528)	Changes the look and feel of your applications.
	DocumentFontColor (see page 3529)	This is DocumentFontColor, a member of class TRibbonLunaColorMap.

	FontColor (see page 3529)	Changes the look and feel of your applications.
	GroupFontColor (see page 3529)	This is GroupFontColor, a member of class TRibbonLunaColorMap.
	HighlightColor (see page 3529)	Changes the look and feel of your applications.
	InactiveTabFontColor (see page 3530)	This is InactiveTabFontColor, a member of class TRibbonLunaColorMap.
	QuickAccessToolbarMoreBorderColor (see page 3530)	This is QuickAccessToolbarMoreBorderColor, a member of class TRibbonLunaColorMap.
	QuickAccessToolbarMoreColor (see page 3530)	This is QuickAccessToolbarMoreColor, a member of class TRibbonLunaColorMap.

62.1.22.2 TRibbonLunaColorMap Methods

The methods of the TRibbonLunaColorMap class are listed here.

Public Methods

	Name	Description
	UpdateColors (see page 3526)	Updates an action component to reflect the current color property.

62.1.22.2.1 TRibbonLunaColorMap.UpdateColors Method

Updates an action component to reflect the current color property.

Pascal

```
procedure UpdateColors; override;
```

C++

```
virtual __fastcall UpdateColors();
```

Description

UpdateColors is called automatically to update associated action components.

62.1.22.3 TRibbonLunaColorMap Properties

The properties of the TRibbonLunaColorMap class are listed here.

Published Properties

	Name	Description
	ActiveTabFontColor (see page 3527)	This is ActiveTabFontColor, a member of class TRibbonLunaColorMap.
	BevelMinimizedInner (see page 3527)	This is BevelMinimizedInner, a member of class TRibbonLunaColorMap.
	BevelMinimizedOuter (see page 3527)	This is BevelMinimizedOuter, a member of class TRibbonLunaColorMap.
	BtnSelectedFont (see page 3527)	Changes the look and feel of your applications.
	CaptionFontColor (see page 3528)	This is CaptionFontColor, a member of class TRibbonLunaColorMap.
	ControlColor (see page 3528)	This is ControlColor, a member of class TRibbonLunaColorMap.
	ControlFrameColor (see page 3528)	This is ControlFrameColor, a member of class TRibbonLunaColorMap.
	ControlInactiveColor (see page 3528)	This is ControlInactiveColor, a member of class TRibbonLunaColorMap.
	DisabledFontColor (see page 3528)	Changes the look and feel of your applications.
	DocumentFontColor (see page 3529)	This is DocumentFontColor, a member of class TRibbonLunaColorMap.
	FontColor (see page 3529)	Changes the look and feel of your applications.
	GroupFontColor (see page 3529)	This is GroupFontColor, a member of class TRibbonLunaColorMap.
	HighlightColor (see page 3529)	Changes the look and feel of your applications.
	InactiveTabFontColor (see page 3530)	This is InactiveTabFontColor, a member of class TRibbonLunaColorMap.
	QuickAccessToolbarMoreBorderColor (see page 3530)	This is QuickAccessToolbarMoreBorderColor, a member of class TRibbonLunaColorMap.

	QuickAccessToolbarMoreColor (see page 3530)	This is QuickAccessToolbarMoreColor, a member of class TRibbonLunaColorMap.

62.1.22.3.1 TRibbonLunaColorMap.ActiveTabFontColor Property

Pascal

```
property ActiveTabFontColor: TColor;
```

C++

```
__property TColor ActiveTabFontColor;
```

Description

This is ActiveTabFontColor, a member of class TRibbonLunaColorMap.

62.1.22.3.2 TRibbonLunaColorMap.BevelMinimizedInner Property

Pascal

```
property BevelMinimizedInner: TColor;
```

C++

```
__property TColor BevelMinimizedInner;
```

Description

This is BevelMinimizedInner, a member of class TRibbonLunaColorMap.

62.1.22.3.3 TRibbonLunaColorMap.BevelMinimizedOuter Property

Pascal

```
property BevelMinimizedOuter: TColor;
```

C++

```
__property TColor BevelMinimizedOuter;
```

Description

This is BevelMinimizedOuter, a member of class TRibbonLunaColorMap.

62.1.22.3.4 TRibbonLunaColorMap.BtnSelectedFont Property

Changes the look and feel of your applications.

Pascal

```
property BtnSelectedFont;
```

C++

```
__property BtnSelectedFont;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.22.3.5 TRibbonLunaColorMap.CaptionFontColor Property

Pascal

```
property CaptionFontColor: TColor;
```

C++

```
__property TColor CaptionFontColor;
```

Description

This is CaptionFontColor, a member of class TRibbonLunaColorMap.

62.1.22.3.6 TRibbonLunaColorMap.ControlColor Property

Pascal

```
property ControlColor: TColor;
```

C++

```
__property TColor ControlColor;
```

Description

This is ControlColor, a member of class TRibbonLunaColorMap.

62.1.22.3.7 TRibbonLunaColorMap.ControlFrameColor Property

Pascal

```
property ControlFrameColor: TColor;
```

C++

```
__property TColor ControlFrameColor;
```

Description

This is ControlFrameColor, a member of class TRibbonLunaColorMap.

62.1.22.3.8 TRibbonLunaColorMap.ControlInactiveColor Property

Pascal

```
property ControlInactiveColor: TColor;
```

C++

```
__property TColor ControlInactiveColor;
```

Description

This is ControlInactiveColor, a member of class TRibbonLunaColorMap.

62.1.22.3.9 TRibbonLunaColorMap.DisabledFontColor Property

Changes the look and feel of your applications.

Pascal

```
property DisabledFontColor;
```

C++

```
__property DisabledFontColor;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.22.3.10 TRibbonLunaColorMap.DocumentFontColor Property

Pascal

```
property DocumentFontColor: TColor;
```

C++

```
__property TColor DocumentFontColor;
```

Description

This is DocumentFontColor, a member of class TRibbonLunaColorMap.

62.1.22.3.11 TRibbonLunaColorMap.FontColor Property

Changes the look and feel of your applications.

Pascal

```
property FontColor;
```

C++

```
__property FontColor;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.22.3.12 TRibbonLunaColorMap.GroupFontColor Property

Pascal

```
property GroupFontColor: TColor;
```

C++

```
__property TColor GroupFontColor;
```

Description

This is GroupFontColor, a member of class TRibbonLunaColorMap.

62.1.22.3.13 TRibbonLunaColorMap.HighlightColor Property

Changes the look and feel of your applications.

Pascal

```
property HighlightColor;
```

C++

```
__property HighlightColor;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.22.3.14 TRibbonLunaColorMap.InactiveTabFontColor Property

Pascal

```
property InactiveTabFontColor: TColor;
```

C++

```
__property TColor InactiveTabFontColor;
```

Description

This is InactiveTabFontColor, a member of class TRibbonLunaColorMap.

62.1.22.3.15 TRibbonLunaColorMap.QuickAccessToolbarMoreBorderColor Property

Pascal

```
property QuickAccessToolbarMoreBorderColor: TColor;
```

C++

```
__property TColor QuickAccessToolbarMoreBorderColor;
```

Description

This is QuickAccessToolbarMoreBorderColor, a member of class TRibbonLunaColorMap.

62.1.22.3.16 TRibbonLunaColorMap.QuickAccessToolbarMoreColor Property

Pascal

```
property QuickAccessToolbarMoreColor: TColor;
```

C++

```
__property TColor QuickAccessToolbarMoreColor;
```

Description

This is QuickAccessToolbarMoreColor, a member of class TRibbonLunaColorMap.

62.1.23 TRibbonObsidianColorMap Class

Color map class for Obsidian.

Class Hierarchy

Pascal

```
TRibbonObsidianColorMap = class(TCustomRibbonColorMap);
```

C++

```
class TRibbonObsidianColorMap : public TCustomRibbonColorMap;
```

File

Ribbon

Description

TRibbonObsidianColorMap is the color map class for the Obsidian ribbon style.

See Also

TRibbonLunaColorMap (see page 3525)

TRibbonSilverColorMap (see page 3549)

62.1.23.1 TRibbonObsidianColorMap Members

The following tables list the members exposed by TRibbonObsidianColorMap.

Public Methods

	Name	Description
	UpdateColors (see page 3532)	Updates an action component to reflect the current color property.

Published Properties

	Name	Description
	ActiveTabFontColor (see page 3532)	This is ActiveTabFontColor, a member of class TRibbonObsidianColorMap.
	BevelMinimizedInner (see page 3533)	This is BevelMinimizedInner, a member of class TRibbonObsidianColorMap.
	BevelMinimizedOuter (see page 3533)	This is BevelMinimizedOuter, a member of class TRibbonObsidianColorMap.
	BtnSelectedFont (see page 3533)	Changes the look and feel of your applications.
	CaptionFontColor (see page 3533)	This is CaptionFontColor, a member of class TRibbonObsidianColorMap.
	ControlColor (see page 3533)	This is ControlColor, a member of class TRibbonObsidianColorMap.
	ControlFrameColor (see page 3534)	This is ControlFrameColor, a member of class TRibbonObsidianColorMap.
	ControlInactiveColor (see page 3534)	This is ControlInactiveColor, a member of class TRibbonObsidianColorMap.
	DisabledFontColor (see page 3534)	Changes the look and feel of your applications.
	DocumentFontColor (see page 3534)	This is DocumentFontColor, a member of class TRibbonObsidianColorMap.
	FontColor (see page 3535)	Changes the look and feel of your applications.
	GroupFontColor (see page 3535)	This is GroupFontColor, a member of class TRibbonObsidianColorMap.
	HighlightColor (see page 3535)	Changes the look and feel of your applications.
	InactiveTabFontColor (see page 3535)	This is InactiveTabFontColor, a member of class TRibbonObsidianColorMap.
	QuickAccessToolbarMoreBorderColor (see page 3536)	This is QuickAccessToolbarMoreBorderColor, a member of class TRibbonObsidianColorMap.
	QuickAccessToolbarMoreColor (see page 3536)	This is QuickAccessToolbarMoreColor, a member of class TRibbonObsidianColorMap.

62.1.23.2 TRibbonObsidianColorMap Methods

The methods of the TRibbonObsidianColorMap class are listed here.

Public Methods

	Name	Description
	UpdateColors (see page 3532)	Updates an action component to reflect the current color property.

62.1.23.2.1 TRibbonObsidianColorMap.UpdateColors Method

Updates an action component to reflect the current color property.

Pascal

```
procedure UpdateColors; override;
```

C++

```
virtual __fastcall UpdateColors();
```

Description

UpdateColors is called automatically to update associated action components.

62.1.23.3 TRibbonObsidianColorMap Properties

The properties of the TRibbonObsidianColorMap class are listed here.

Published Properties

	Name	Description
	ActiveTabFontColor (see page 3532)	This is ActiveTabFontColor, a member of class TRibbonObsidianColorMap.
	BevelMinimizedInner (see page 3533)	This is BevelMinimizedInner, a member of class TRibbonObsidianColorMap.
	BevelMinimizedOuter (see page 3533)	This is BevelMinimizedOuter, a member of class TRibbonObsidianColorMap.
	BtnSelectedFont (see page 3533)	Changes the look and feel of your applications.
	CaptionFontColor (see page 3533)	This is CaptionFontColor, a member of class TRibbonObsidianColorMap.
	ControlColor (see page 3533)	This is ControlColor, a member of class TRibbonObsidianColorMap.
	ControlFrameColor (see page 3534)	This is ControlFrameColor, a member of class TRibbonObsidianColorMap.
	ControlInactiveColor (see page 3534)	This is ControlInactiveColor, a member of class TRibbonObsidianColorMap.
	DisabledFontColor (see page 3534)	Changes the look and feel of your applications.
	DocumentFontColor (see page 3534)	This is DocumentFontColor, a member of class TRibbonObsidianColorMap.
	FontColor (see page 3535)	Changes the look and feel of your applications.
	GroupFontColor (see page 3535)	This is GroupFontColor, a member of class TRibbonObsidianColorMap.
	HighlightColor (see page 3535)	Changes the look and feel of your applications.
	InactiveTabFontColor (see page 3535)	This is InactiveTabFontColor, a member of class TRibbonObsidianColorMap.
	QuickAccessToolbarMoreBorderColor (see page 3536)	This is QuickAccessToolbarMoreBorderColor, a member of class TRibbonObsidianColorMap.
	QuickAccessToolbarMoreColor (see page 3536)	This is QuickAccessToolbarMoreColor, a member of class TRibbonObsidianColorMap.

62.1.23.3.1 TRibbonObsidianColorMap.ActiveTabFontColor Property

Pascal

```
property ActiveTabFontColor: TColor;
```

C++

```
__property TColor ActiveTabFontColor;
```

Description

This is ActiveTabFontColor, a member of class TRibbonObsidianColorMap.

62.1.23.3.2 TRibbonObsidianColorMap.BevelMinimizedInner Property

Pascal

```
property BevelMinimizedInner: TColor;
```

C++

```
__property TColor BevelMinimizedInner;
```

Description

This is BevelMinimizedInner, a member of class TRibbonObsidianColorMap.

62.1.23.3.3 TRibbonObsidianColorMap.BevelMinimizedOuter Property

Pascal

```
property BevelMinimizedOuter: TColor;
```

C++

```
__property TColor BevelMinimizedOuter;
```

Description

This is BevelMinimizedOuter, a member of class TRibbonObsidianColorMap.

62.1.23.3.4 TRibbonObsidianColorMap.BtnSelectedFont Property

Changes the look and feel of your applications.

Pascal

```
property BtnSelectedFont;
```

C++

```
__property BtnSelectedFont;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.23.3.5 TRibbonObsidianColorMap.CaptionFontColor Property

Pascal

```
property CaptionFontColor: TColor;
```

C++

```
__property TColor CaptionFontColor;
```

Description

This is CaptionFontColor, a member of class TRibbonObsidianColorMap.

62.1.23.3.6 TRibbonObsidianColorMap.ControlColor Property

Pascal

```
property ControlColor: TColor;
```

C++

```
__property TColor ControlColor;
```

Description

This is ControlColor, a member of class TRibbonObsidianColorMap.

62.1.23.3.7 TRibbonObsidianColorMap.ControlFrameColor Property

Pascal

```
property ControlFrameColor: TColor;
```

C++

```
__property TColor ControlFrameColor;
```

Description

This is ControlFrameColor, a member of class TRibbonObsidianColorMap.

62.1.23.3.8 TRibbonObsidianColorMap.ControlInactiveColor Property

Pascal

```
property ControlInactiveColor: TColor;
```

C++

```
__property TColor ControlInactiveColor;
```

Description

This is ControlInactiveColor, a member of class TRibbonObsidianColorMap.

62.1.23.3.9 TRibbonObsidianColorMap.DisabledFontColor Property

Changes the look and feel of your applications.

Pascal

```
property DisabledFontColor;
```

C++

```
__property DisabledFontColor;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.23.3.10 TRibbonObsidianColorMap.DocumentFontColor Property

Pascal

```
property DocumentFontColor: TColor;
```

C++

```
__property TColor DocumentFontColor;
```

Description

This is DocumentFontColor, a member of class TRibbonObsidianColorMap.

62.1.23.3.11 TRibbonObsidianColorMap.FontColor Property

Changes the look and feel of your applications.

Pascal

```
property FontColor;
```

C++

```
__property FontColor;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.23.3.12 TRibbonObsidianColorMap.GroupFontColor Property

Pascal

```
property GroupFontColor: TColor;
```

C++

```
__property TColor GroupFontColor;
```

Description

This is GroupFontColor, a member of class TRibbonObsidianColorMap.

62.1.23.3.13 TRibbonObsidianColorMap.HighlightColor Property

Changes the look and feel of your applications.

Pascal

```
property HighlightColor;
```

C++

```
__property HighlightColor;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.23.3.14 TRibbonObsidianColorMap.InactiveTabFontColor Property

Pascal

```
property InactiveTabFontColor: TColor;
```

C++

```
__property TColor InactiveTabFontColor;
```

Description

This is InactiveTabFontColor, a member of class TRibbonObsidianColorMap.

62.1.23.3.15 TRibbonObsidianColorMap.QuickAccessToolbarMoreBorderColor Property

Pascal

```
property QuickAccessToolbarMoreBorderColor: TColor;
```

C++

```
__property TColor QuickAccessToolbarMoreBorderColor;
```

Description

This is QuickAccessToolbarMoreBorderColor, a member of class TRibbonObsidianColorMap.

62.1.23.3.16 TRibbonObsidianColorMap.QuickAccessToolbarMoreColor Property

Pascal

```
property QuickAccessToolbarMoreColor: TColor;
```

C++

```
__property TColor QuickAccessToolbarMoreColor;
```

Description

This is QuickAccessToolbarMoreColor, a member of class TRibbonObsidianColorMap.

62.1.24 TRibbonPage Class

Tab page for ribbon control.

Class Hierarchy

Pascal

```
TRibbonPage = class(TCustomRibbonPage);
```

C++

```
class TRibbonPage : public TCustomRibbonPage;
```

File

Ribbon

Description

TRibbonPage is a TCustomRibbonPage (see page 3425) tab page for the ribbon control that holds the groups of controls displayed.

See Also

Creating an Application that Uses Ribbon Controls

Adding Commands to the Ribbon

TCustomRibbonPage (see page 3425)

62.1.24.1 TRibbonPage Members

The following tables list the members exposed by TRibbonPage.

Protected Methods

	Name	Description
	CMControlListChanging (see page 3429)	This is CMControlListChanging, a member of class TCustomRibbonPage.
	CMTextchanged (see page 3429)	This is CMTextchanged, a member of class TCustomRibbonPage.
V	CreateScrollButton (see page 3430)	This is CreateScrollButton, a member of class TCustomRibbonPage.
V	DoScrollButtonClick (see page 3430)	This is DoScrollButtonClick, a member of class TCustomRibbonPage.
	GetPageHeight (see page 3431)	This is GetPageHeight, a member of class TCustomRibbonPage.
	GetPageTop (see page 3431)	This is GetPageTop, a member of class TCustomRibbonPage.
V	Loaded (see page 3432)	Initializes the control after it is loaded from a stream.
V	Notification (see page 3433)	Responds to notifications that components are being created or destroyed.
V	Paint (see page 3433)	Renders the image of a custom control.
V	RequestAlign (see page 3434)	Instructs the parent of a control to reposition the control, enforcing its Align property.
V	Resize (see page 3434)	Respond to control resize.
V	SetParent (see page 3435)	Sets the parent of the control.
	UpdateScrollButtons (see page 3436)	This is UpdateScrollButtons, a member of class TCustomRibbonPage.

Protected Properties

	Name	Description
	GroupOffset (see page 3437)	This is GroupOffset, a member of class TCustomRibbonPage.

Public Constants

	Name	Description
	cRibbonGroupLeftOffset (see page 3428)	This is cRibbonGroupLeftOffset, a member of class TCustomRibbonPage.
	cRibbonGroupSpacing (see page 3428)	This is cRibbonGroupSpacing, a member of class TCustomRibbonPage.

Public Methods

	Name	Description
	AddGroup (see page 3429)	This is AddGroup, a member of class TCustomRibbonPage.
	AlignGroups (see page 3429)	This is AlignGroups, a member of class TCustomRibbonPage.
	CollapseGroup (see page 3430)	This is CollapseGroup, a member of class TCustomRibbonPage.
V	Create (see page 3426)	Creates an instance of TCustomControl.
V	Destroy (see page 3427)	Destroys an instance of TCustomControl.
	FindGroup (see page 3430)	This is FindGroup, a member of class TCustomRibbonPage.
	FindGroupAt (see page 3430)	This is FindGroupAt, a member of class TCustomRibbonPage.
	GetGroupPos (see page 3431)	This is GetGroupPos, a member of class TCustomRibbonPage.
	GetNextVisibleGroup (see page 3431)	This is GetNextVisibleGroup, a member of class TCustomRibbonPage.
	HandleKeyTips (see page 3432)	This is HandleKeyTips, a member of class TCustomRibbonPage.
	HideKeyTips (see page 3432)	This is HideKeyTips, a member of class TCustomRibbonPage.
V	Invalidate (see page 3432)	Schedules a control repaint.
	RibbonStyle (see page 3434)	Return style class to use when drawing specific Ribbon (see page 3438) elements.
V	SetBounds (see page 3435)	Sets the windowed control's boundary properties all at once.
	ShowKeyTips (see page 3436)	This is ShowKeyTips, a member of class TCustomRibbonPage.
	UpdateActionManager (see page 3436)	This is UpdateActionManager, a member of class TCustomRibbonPage.

Public Properties

	Name	Description
R	CollapseGroupCount (see page 3437)	This is CollapseGroupCount, a member of class TCustomRibbonPage.
R	GroupCount (see page 3437)	This is GroupCount, a member of class TCustomRibbonPage.

R	Groups (see page 3437)	This is Groups, a member of class TCustomRibbonPage.
	Ribbon (see page 3438)	This is Ribbon, a member of class TCustomRibbonPage.

Published Properties

	Name	Description
	BiDiMode (see page 3538)	Specifies the bi-directional mode for the control.
	Caption (see page 3539)	Specifies a text string that identifies the control to the user.
	Height (see page 3541)	Specifies the vertical size of the control in pixels.
	Index (see page 3542)	This is Index, a member of class TRibbonPage.
	KeyTip (see page 3543)	This is KeyTip, a member of class TRibbonPage.
	Name (see page 3543)	Specifies the name of the component as referenced in code.
	ParentBiDiMode (see page 3545)	Specifies whether the control uses its parent's BiDiMode.

62.1.24.2 TRibbonPage Properties

The properties of the TRibbonPage class are listed here.

Published Properties

	Name	Description
	BiDiMode (see page 3538)	Specifies the bi-directional mode for the control.
	Caption (see page 3539)	Specifies a text string that identifies the control to the user.
	Height (see page 3541)	Specifies the vertical size of the control in pixels.
	Index (see page 3542)	This is Index, a member of class TRibbonPage.
	KeyTip (see page 3543)	This is KeyTip, a member of class TRibbonPage.
	Name (see page 3543)	Specifies the name of the component as referenced in code.
	ParentBiDiMode (see page 3545)	Specifies whether the control uses its parent's BiDiMode.

62.1.24.2.1 TRibbonPage.BiDiMode Property

Specifies the bi-directional mode for the control.

Pascal

```
property BiDiMode;
```

C++

```
__property BiDiMode;
```

Description

Use BiDiMode to enable the control to adjust its appearance and behavior automatically when the application runs in a locale that reads from right to left instead of left to right. The bi-directional mode controls the reading order for the text, the placement of the vertical scroll bar, and whether the alignment is changed.

Alignment does not change for controls that are known to contain number, date, time, or currency values. For example, with data aware controls, the alignment does not change for the following field types: ftSmallint, ftInteger, ftWord, ftFloat, ftCurrency, ftBCD, ftDate, ftTime, ftDateTime, ftAutoInc.

See Also

UseRightToLeftAlignment

UseRightToLeftReading

UseRightToLeftScrollBar

Including Bi-directional Functionality in Applications

62.1.24.2.2 TRibbonPage.Caption Property

Specifies a text string that identifies the control to the user.

Pascal

```
property Caption;
```

C++

```
__property Caption;
```

Description

Use Caption to specify the text string that labels the control.

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

Note: Controls that display text use either the Caption property or the Text property to specify the text value. Which property is used depends on the type of control. In general, Caption is used for text that appears as a window title or label, while Text is used for text that appears as the content of a control.

Note: The Caption property inheritted from TControl is a protected property. Access to the Caption requires casting the class to an inheritted one where Caption is a published or public property.

C++ Examples:

```
/*
The following code keeps the size of the label control
constant, even though the length of the label's caption
changes. As a result, the caption of the label is probably
too long to display in the label when the user clicks the
button:
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Label1->AutoSize = false;
  Label1->Caption = "This string is too long to be the caption of this label";
}
/*
This example requires a TPageControl populated by several
TabSheets, a TEdit and a TButton. When the button is
clicked, the caption of the active tab changes to that of
the TEdit.  These lines of code could also be written:
PageControl1.ActivePage.Caption := Edit1.Text;
This would make a TTabSheet variable unnecessary since
PageControl1.->ActivePage is already of type TTabSheet.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TTabSheet *sheet = PageControl1->ActivePage;
  sheet->Caption = Edit1->Text;
}
/*
This example creates a ToolBar and adds ToolButtons to it.
Include ComCtlrs in the uses clause.
*/
#include <comctrls.hpp>
#include <memory>
```

```
void AddButtons(TToolBar *toolBar, TStringList *captions)
{
  for (int i = 0; i < captions->Count; i++)
  {
    TToolButton *button = new TToolButton(toolBar);  // The owner (toolBar) will clean this up.
    button->Parent = toolBar;
    button->Caption = captions->Strings[i];
    if (button->Caption == "|")
      button->Style = tbsSeparator;
    else
      button->Style = tbsButton;
  }
}

TToolBar *myToolBar;

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  myToolBar = new TToolBar(this); // The owner (this) will clean this up.
  myToolBar->Parent = this;
//  TStringList *captions = new TStringList();
  std::auto_ptr<TStringList> captions(new TStringList());

  captions->Add("New");
  captions->Add("Save");
  captions->Add("|");
  captions->Add("Cut");
  captions->Add("Copy");
  captions->Add("Paste");
  AddButtons(myToolBar, captions.get());
//  delete captions;
  myToolBar->ShowCaptions = True;
  myToolBar->Height = 40;
}
```

Delphi Examples:

```
{
The following code keeps the size of the label control
constant, even though the length of the label's caption
changes. As a result, the caption of the label is probably
too long to display in the label when the user clicks the
button:
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.AutoSize := False;
  Label1.Caption :=
    'This string is too long to be the caption of this label';
end;
{
This example requires a TPageControl populated by several
TabSheets, a TEdit and a TButton. When the button is
clicked, the caption of the active tab changes to that of
the TEdit.  These lines of code could also be written:
PageControl1.ActivePage.Caption := Edit1.Text;
This would obviate the need for a TTabSheet variable since
PageControl1.->ActivePage is already of type TTabSheet.
}
procedure TForm1.Button1Click(Sender: TObject);
var
  sheet: TTabSheet;
begin
  sheet := PageControl1.ActivePage;
  sheet.Caption := Edit1.Text;
end;
```

```
{
This example creates a ToolBar and adds ToolButtons to it.
Include ComCtlrs in the uses clause.
}
procedure AddButtons(ToolBar: TToolBar;
  Const ButtonCaptions: array of String);
var
  I: Integer;
begin
  for I := 0 to High(ButtonCaptions) do
    begin
      with TToolButton.Create(ToolBar) do
      begin
        Parent := ToolBar;
        Caption := ButtonCaptions[I];
        if (ButtonCaptions[I] = '|') then
          Style := tbsSeparator
        else
          Style := tbsButton;
      end;
    end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1 := TToolBar.Create(Self);
  ToolBar1.Parent := Self;
  AddButtons(ToolBar1, ['New', 'Save', '|', 'Cut', 'Copy', 'Paste']);
  ToolBar1.ShowCaptions := True;
  ToolBar1.Height := 40;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  I: Integer;
begin
  for I := ToolBar1.ButtonCount - 1 downto 0 do
    ToolBar1.Buttons[I].Free;
  ToolBar1.Free;
end;
```

62.1.24.2.3 TRibbonPage.Height Property

Specifies the vertical size of the control in pixels.

Pascal

```
property Height;
```

C++

```
__property Height;
```

Description

Use the Height property to read or change the height of the control.

C++ Examples:

```
/*
This example needs only a blank form.  Three Rich Edit
controls are placed vertically on the form, each contain a
line of text, and the second form has HideSelection set to
false.  Try tabbing between the three Rich Edit controls and
observe the effect on the selected text.
*/
#include <ComCtrls.hpp>
```

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int I = 0; I <= 2; I++)
  {
    TRichEdit *pRich = new TRichEdit(this);
    pRich->Parent = this;
    pRich->Top = pRich->Height * I;
    pRich->Text = "This text will take the focus.";
    pRich->SelStart = 0;
    pRich->SelLength = pRich->Text.Length();
    if (I == 1)
      pRich->HideSelection = false;
  }
}
```

Delphi Examples:

```
{
This example needs only a blank form. Three Rich Edit
controls are placed vertically on the form, each contain a
line of text, and the second form has HideSelection set to
false.  Try tabbing between the three Rich Edit controls and
observe the effect on the selected text.
}
uses ComCtrls;

procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
  for I := 0 to 2 do
    with TRichEdit.Create(Self) do
    begin
      Parent := Self;
      Top := Height * I;
      Text := 'This text will take the focus.';
      SelStart := 0;
      SelLength := Length(Text);
      if (I = 1) then
        HideSelection := False;
    end;
end;
```

See Also

ClientHeight

Width

Using TDBListBox and TDBComboBox

62.1.24.2.4 TRibbonPage.Index Property

Pascal

```
property Index: Integer;
```

C++

```
__property int Index;
```

Description

This is Index, a member of class TRibbonPage.

62.1.24.2.5 TRibbonPage.KeyTip Property

Pascal

```
property KeyTip: string;
```

C++

```
__property UnicodeString KeyTip;
```

Description

This is KeyTip, a member of class TRibbonPage.

62.1.24.2.6 TRibbonPage.Name Property

Specifies the name of the component as referenced in code.

Pascal

```
property Name;
```

C++

```
__property Name;
```

Description

Use Name to change the name of a component to reflect its purpose in the current application. By default, the IDE assigns sequential names based on the type of the component, such as 'Button1', 'Button2', and so on.

Use Name to refer to the component in code.

Warning: Changing Name at runtime causes any references to the old name to become undefined. Any subsequent code that uses the old name will cause an exception.

C++ Examples:

```
/*
For the following example, add a button, a status bar, and a
list box to the form. Set the SimplePanel property of the
status bar to true, using the object inspector. Also, populate
the OnMouseUp event handler for the list box.  The following
code fills a list box with the names of all components on the
form when the user clicks the button. References to the
components themselves are inserted along with the names. The
components are all inserted at the front of the list, so that
the last component added to the form is the first component
in the list.  When the user right-clicks the name of an
object in the list, the component's coordinates are displayed
on the status bar. Note that because we are using the right-
click, the item need not be selected.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  for (int i = 0; i < ComponentCount; i++)
    ListBox1->Items->InsertObject(0,
                                  Components[i]->Name,
                                  dynamic_cast<TObject *>(Components[i]));
}

void __fastcall TForm1::ListBox1MouseUp(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
```

```
  if (Button == mbRight)
  {
    TClass ClassRef;
    int Index = ListBox1->ItemAtPos(Point(X,Y), true);
    // only components that are controls have a position
    // make sure the component is a control
    for (ClassRef = ListBox1->Items->Objects[Index]->ClassType();
         ClassRef != NULL;
         ClassRef = ClassRef->ClassParent())
      if (String(ClassRef->ClassName()) == "TControl")
      {
        TControl *TheObject = dynamic_cast<TControl *>(ListBox1->Items->Objects[Index]);
        StatusBar1->SimpleText =
          TheObject->Name + " is at (" +
          IntToStr(TheObject->Left) + ", " +
          IntToStr(TheObject->Top) + ")";
        break;
      }
    if (ClassRef == NULL) // if it wasn't a control
      MessageBeep(0);
  }
}
```

Delphi Examples:

```
{
For the following example, add a button, a status bar, and a
list box to the form. Set the SimplePanel property of the
status bar to true, using the object inspector. Also, populate
the OnMouseUp event handler for the list box.  The following
code fills a list box with the names of all components on the
form when the user clicks the button. References to the
components themselves are inserted along with the names. The
components are all inserted at the front of the list, so that
the last component added to the form is the first component
in the list.  When the user right-clicks the name of an
object in the list, the component's coordinates are displayed
on the status bar. Note that because we are using the right-
click, the item need not be selected.
}
procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
begin
  for I := 0 to Form1.ComponentCount-1 do
    ListBox1.Items.InsertObject(
      0, Form1.Components[I].Name,
      Form1.Components[I] as TObject);
end;

procedure TForm1.ListBox1MouseUp(
  Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X,Y: integer);
var
  APoint: TPoint;
  Index: integer;
  TheObject: TControl;
begin
  if Button = mbRight then
  begin
  APoint.X := X;
  APoint.Y := Y;
  Index := ListBox1.ItemAtPos(APoint, True);
  if (ListBox1.Items.Objects[Index] is TControl) then
  begin
    TheObject :=
      (ListBox1.Items.Objects[Index] as TControl);
```

```
      StatusBar1.SimpleText := TheObject.Name + ' is at (' +
                               IntToStr(TheObject.Left) + ', ' +
                               IntToStr(TheObject.Top) + ') ';
    end
    else
      Beep;
    end;
  end;
```

See Also

ChangeName

SetName

ValidateRename

Defining New Persistent Fields

Using TSQLQuery

Using TSQLStoredProc

Using TSQLTable

62.1.24.2.7 TRibbonPage.ParentBiDiMode Property

Specifies whether the control uses its parent's BiDiMode.

Pascal

```
property ParentBiDiMode;
```

C++

```
__property ParentBiDiMode;
```

Description

Use ParentBiDiMode to get or set whether the control uses its parent's BiDiMode. When ParentBiDiMode is true, the control's BiDiMode property is always equal to that of its parent. If ParentBiDiMode is false, the control uses its own BiDiMode property.

Set ParentBiDiMode to true for all controls in a form in order to ensure that the entire user interface adjusts to Middle Eastern locales in a uniform manner.

When the value of the control's BiDiMode property changes, ParentBiDiMode becomes false automatically.

See Also

Parent

BiDiMode

62.1.25 TRibbonQuickAccessPopupToolbar Class

Class Hierarchy

Pascal

```
TRibbonQuickAccessPopupToolbar = class(TCustomRibbonQuickAccessPopupToolbar);
```

C++

```
class TRibbonQuickAccessPopupToolbar : public TCustomRibbonQuickAccessPopupToolbar;
```

File

Ribbon

Description

This is class Ribbon.TRibbonQuickAccessPopupToolbar.

62.1.26 TRibbonQuickAccessToolbar Class

Quck Access Toolbar.

Class Hierarchy

Pascal

```
TRibbonQuickAccessToolbar = class(TCustomRibbonQuickAccessToolbar);
```

C++

```
class TRibbonQuickAccessToolbar : public TCustomRibbonQuickAccessToolbar;
```

File

Ribbon

Description

TRibbonQuickAccessToolbar is a TCustomRibbonQuickAccessToolbar (see page 3441) Quck Access Toolbar implementation.

See Also

Creating an Application that Uses Ribbon Controls

Adding Commands to the Ribbon

TCustomRibbonQuickAccessToolbar (see page 3441)

62.1.26.1 TRibbonQuickAccessToolbar Members

The following tables list the members exposed by TRibbonQuickAccessToolbar.

Protected Methods

	Name	Description
	CMFontchanged (see page 3445)	This is CMFontchanged, a member of class TCustomRibbonQuickAccessToolbar.
	CreateControls (see page 3446)	This is CreateControls, a member of class TCustomRibbonQuickAccessToolbar.
	DoContextPopup (see page 3446)	OnContextPopup event dispatcher.
	DoDropActions (see page 3446)	This is DoDropActions, a member of class TCustomRibbonQuickAccessToolbar.
	DoDropItem (see page 3447)	This is DoDropItem, a member of class TCustomRibbonQuickAccessToolbar.
	DoMoreCommands (see page 3447)	This is DoMoreCommands, a member of class TCustomRibbonQuickAccessToolbar.

	DoScrollBtnClick (see page 3447)	This is DoScrollBtnClick, a member of class TCustomRibbonQuickAccessToolbar.
	DrawBackground (see page 3448)	This is DrawBackground, a member of class TCustomRibbonQuickAccessToolbar.
	GetBarHeight (see page 3448)	This is GetBarHeight, a member of class TCustomRibbonQuickAccessToolbar.
	GetBarWidth (see page 3449)	This is GetBarWidth, a member of class TCustomRibbonQuickAccessToolbar.
	GetScrollBtnClass (see page 3449)	This is GetScrollBtnClass, a member of class TCustomRibbonQuickAccessToolbar.
	Notification (see page 3449)	Responds to notifications that components are being created or destroyed.
	RequestAlign (see page 3450)	Instructs the parent of a control to reposition the control, enforcing its Align property.
	SetParent (see page 3450)	Sets the parent of the control.
	SetupDropDownBtn (see page 3451)	This is SetupDropDownBtn, a member of class TCustomRibbonQuickAccessToolbar.
	ShowCustomizeMenu (see page 3451)	This is ShowCustomizeMenu, a member of class TCustomRibbonQuickAccessToolbar.
	ShowMoreControls (see page 3451)	This is ShowMoreControls, a member of class TCustomRibbonQuickAccessToolbar.
	UpdateQATWidth (see page 3451)	This is UpdateQATWidth, a member of class TCustomRibbonQuickAccessToolbar.

Public Methods

	Name	Description
	BarHandleKeyTips (see page 3444)	Check if any commands on Action Bar are partial or full match for **KeyTip**.
	BarHandleKeyTips (see page 3913)	Check if any commands on Action Bar are partial or full match for **KeyTip**.
	BarHideKeyTips (see page 3444)	Hide any visible **KeyTip** windows.
	BarHideKeyTips (see page 3913)	Hide any visible **KeyTip** windows.
	BarShowKeyTips (see page 3445)	Show **KeyTip**s for any commands that support them.
	BarShowKeyTips (see page 3914)	Show **KeyTip**s for any commands that support them.
	Create (see page 3443)	Creates and initializes a TCustomActionToolBar instance.
	CreateControl (see page 3445)	This is CreateControl, a member of class TCustomRibbonQuickAccessToolbar.
	Customize (see page 3446)	This is Customize, a member of class TCustomRibbonQuickAccessToolbar.
	DoDropCategory (see page 3447)	This is DoDropCategory, a member of class TCustomRibbonQuickAccessToolbar.
	DoUpdateQATWidth (see page 3570)	This is DoUpdateQATWidth, a member of class IRibbonQuickAccessToolbar.
	DoUpdateQATWidth (see page 3448)	This is DoUpdateQATWidth, a member of class TCustomRibbonQuickAccessToolbar.
	DragDrop (see page 3448)	Generates an OnDragDrop event.

Public Nested Types

Name	Description
TMoreCommandsEvent (see page 3453)	This is nested type Ribbon.TCustomRibbonQuickAccessToolbar.TMoreCommandsEvent.

Public Properties

	Name	Description
	Items (see page 3452)	Lists the action clients that correspond to the child controls of this action band.
	Ribbon (see page 3452)	This is Ribbon, a member of class TCustomRibbonQuickAccessToolbar.
	ScrollMode (see page 3453)	This is ScrollMode, a member of class TCustomRibbonQuickAccessToolbar.
	ShowBorder (see page 3453)	This is ShowBorder, a member of class TCustomRibbonQuickAccessToolbar.

Published Properties

	Name	Description
	ActionManager (see page 3548)	Specifies the action manager component that is associated with the action band.
	OnMoreCommands (see page 3548)	This is OnMoreCommands, a member of class TRibbonQuickAccessToolbar.
	VertMargin (see page 3548)	Specifies the width of the vertical margins in pixels.

62.1.26.2 TRibbonQuickAccessToolbar Properties

The properties of the TRibbonQuickAccessToolbar class are listed here.

Published Properties

	Name	Description
	ActionManager (see page 3548)	Specifies the action manager component that is associated with the action band.
	OnMoreCommands (see page 3548)	This is OnMoreCommands, a member of class TRibbonQuickAccessToolbar.
	VertMargin (see page 3548)	Specifies the width of the vertical margins in pixels.

62.1.26.2.1 TRibbonQuickAccessToolbar.ActionManager Property

Specifies the action manager component that is associated with the action band.

Pascal

```
property ActionManager;
```

C++

```
__property ActionManager;
```

Description

Use ActionManager to access the action manager that uses this action band to render UI elements it manages. Typically, this association is created when you drag actions from the action manager and drop them on the action bar. When an action is dropped on an action band and the ActionManager property is not assigned, the ActionManager property is automatically assigned to the action manager that manages the action that was dropped on the action band.

At runtime, you can perform the equivalent operation to dragging an action onto the action band by setting the ActionBar property of one of the action clients that the action manager's ActionBars property lists so that it specifies this component.

See Also

ActionBars

ActionClient

ActionBar

62.1.26.2.2 TRibbonQuickAccessToolbar.OnMoreCommands Property

Pascal

```
property OnMoreCommands: TMoreCommandsEvent;
```

C++

```
__property TMoreCommandsEvent OnMoreCommands;
```

Description

This is OnMoreCommands, a member of class TRibbonQuickAccessToolbar.

62.1.26.2.3 TRibbonQuickAccessToolbar.VertMargin Property

Specifies the width of the vertical margins in pixels.

Pascal

```
property VertMargin;
```

C++

```
__property VertMargin;
```

Description

VertMargin specifies the width of the action band's top and bottom vertical margins in pixels.

See Also

HorzMargin

HorzSeparator

62.1.27 TRibbonSilverColorMap Class

Color map class for Silver.

Class Hierarchy

Pascal

```
TRibbonSilverColorMap = class(TCustomRibbonColorMap);
```

C++

```
class TRibbonSilverColorMap : public TCustomRibbonColorMap;
```

File

Ribbon

Description

TRibbonSilverColorMap is the color map class for the Silver ribbon style.

See Also

TRibbonLunaColorMap (see page 3525)

TRibbonObsidianColorMap (see page 3530)

62.1.27.1 TRibbonSilverColorMap Members

The following tables list the members exposed by TRibbonSilverColorMap.

Public Methods

	Name	Description
	UpdateColors (see page 3550)	Updates an action component to reflect the current color property.

Published Properties

	Name	Description
	ActiveTabFontColor (see page 3551)	This is ActiveTabFontColor, a member of class TRibbonSilverColorMap.
	BevelMinimizedInner (see page 3551)	This is BevelMinimizedInner, a member of class TRibbonSilverColorMap.
	BevelMinimizedOuter (see page 3551)	This is BevelMinimizedOuter, a member of class TRibbonSilverColorMap.

	BtnSelectedFont (see page 3551)	Changes the look and feel of your applications.
	CaptionFontColor (see page 3552)	This is CaptionFontColor, a member of class TRibbonSilverColorMap.
	ControlColor (see page 3552)	This is ControlColor, a member of class TRibbonSilverColorMap.
	ControlFrameColor (see page 3552)	This is ControlFrameColor, a member of class TRibbonSilverColorMap.
	ControlInactiveColor (see page 3552)	This is ControlInactiveColor, a member of class TRibbonSilverColorMap.
	DisabledFontColor (see page 3553)	Changes the look and feel of your applications.
	DocumentFontColor (see page 3553)	This is DocumentFontColor, a member of class TRibbonSilverColorMap.
	FontColor (see page 3553)	Changes the look and feel of your applications.
	GroupFontColor (see page 3553)	This is GroupFontColor, a member of class TRibbonSilverColorMap.
	HighlightColor (see page 3554)	Changes the look and feel of your applications.
	InactiveTabFontColor (see page 3554)	This is InactiveTabFontColor, a member of class TRibbonSilverColorMap.
	QuickAccessToolbarMoreBorderColor (see page 3554)	This is QuickAccessToolbarMoreBorderColor, a member of class TRibbonSilverColorMap.
	QuickAccessToolbarMoreColor (see page 3554)	This is QuickAccessToolbarMoreColor, a member of class TRibbonSilverColorMap.

62.1.27.2 TRibbonSilverColorMap Methods

The methods of the TRibbonSilverColorMap class are listed here.

Public Methods

	Name	Description
	UpdateColors (see page 3550)	Updates an action component to reflect the current color property.

62.1.27.2.1 TRibbonSilverColorMap.UpdateColors Method

Updates an action component to reflect the current color property.

Pascal

```
procedure UpdateColors; override;
```

C++

```
virtual __fastcall UpdateColors();
```

Description

UpdateColors is called automatically to update associated action components.

62.1.27.3 TRibbonSilverColorMap Properties

The properties of the TRibbonSilverColorMap class are listed here.

Published Properties

	Name	Description
	ActiveTabFontColor (see page 3551)	This is ActiveTabFontColor, a member of class TRibbonSilverColorMap.
	BevelMinimizedInner (see page 3551)	This is BevelMinimizedInner, a member of class TRibbonSilverColorMap.
	BevelMinimizedOuter (see page 3551)	This is BevelMinimizedOuter, a member of class TRibbonSilverColorMap.
	BtnSelectedFont (see page 3551)	Changes the look and feel of your applications.
	CaptionFontColor (see page 3552)	This is CaptionFontColor, a member of class TRibbonSilverColorMap.
	ControlColor (see page 3552)	This is ControlColor, a member of class TRibbonSilverColorMap.
	ControlFrameColor (see page 3552)	This is ControlFrameColor, a member of class TRibbonSilverColorMap.
	ControlInactiveColor (see page 3552)	This is ControlInactiveColor, a member of class TRibbonSilverColorMap.
	DisabledFontColor (see page 3553)	Changes the look and feel of your applications.

	DocumentFontColor (see page 3553)	This is DocumentFontColor, a member of class TRibbonSilverColorMap.
	FontColor (see page 3553)	Changes the look and feel of your applications.
	GroupFontColor (see page 3553)	This is GroupFontColor, a member of class TRibbonSilverColorMap.
	HighlightColor (see page 3554)	Changes the look and feel of your applications.
	InactiveTabFontColor (see page 3554)	This is InactiveTabFontColor, a member of class TRibbonSilverColorMap.
	QuickAccessToolbarMoreBorderColor (see page 3554)	This is QuickAccessToolbarMoreBorderColor, a member of class TRibbonSilverColorMap.
	QuickAccessToolbarMoreColor (see page 3554)	This is QuickAccessToolbarMoreColor, a member of class TRibbonSilverColorMap.

62.1.27.3.1 TRibbonSilverColorMap.ActiveTabFontColor Property

Pascal

```
property ActiveTabFontColor: TColor;
```

C++

```
__property TColor ActiveTabFontColor;
```

Description

This is ActiveTabFontColor, a member of class TRibbonSilverColorMap.

62.1.27.3.2 TRibbonSilverColorMap.BevelMinimizedInner Property

Pascal

```
property BevelMinimizedInner: TColor;
```

C++

```
__property TColor BevelMinimizedInner;
```

Description

This is BevelMinimizedInner, a member of class TRibbonSilverColorMap.

62.1.27.3.3 TRibbonSilverColorMap.BevelMinimizedOuter Property

Pascal

```
property BevelMinimizedOuter: TColor;
```

C++

```
__property TColor BevelMinimizedOuter;
```

Description

This is BevelMinimizedOuter, a member of class TRibbonSilverColorMap.

62.1.27.3.4 TRibbonSilverColorMap.BtnSelectedFont Property

Changes the look and feel of your applications.

Pascal

```
property BtnSelectedFont;
```

C++

```
__property BtnSelectedFont;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.27.3.5 TRibbonSilverColorMap.CaptionFontColor Property

Pascal

```
property CaptionFontColor: TColor;
```

C++

```
__property TColor CaptionFontColor;
```

Description

This is CaptionFontColor, a member of class TRibbonSilverColorMap.

62.1.27.3.6 TRibbonSilverColorMap.ControlColor Property

Pascal

```
property ControlColor: TColor;
```

C++

```
__property TColor ControlColor;
```

Description

This is ControlColor, a member of class TRibbonSilverColorMap.

62.1.27.3.7 TRibbonSilverColorMap.ControlFrameColor Property

Pascal

```
property ControlFrameColor: TColor;
```

C++

```
__property TColor ControlFrameColor;
```

Description

This is ControlFrameColor, a member of class TRibbonSilverColorMap.

62.1.27.3.8 TRibbonSilverColorMap.ControlInactiveColor Property

Pascal

```
property ControlInactiveColor: TColor;
```

C++

```
__property TColor ControlInactiveColor;
```

Description

This is ControlInactiveColor, a member of class TRibbonSilverColorMap.

62.1.27.3.9 TRibbonSilverColorMap.DisabledFontColor Property

Changes the look and feel of your applications.

Pascal

```
property DisabledFontColor;
```

C++

```
__property DisabledFontColor;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.27.3.10 TRibbonSilverColorMap.DocumentFontColor Property

Pascal

```
property DocumentFontColor: TColor;
```

C++

```
__property TColor DocumentFontColor;
```

Description

This is DocumentFontColor, a member of class TRibbonSilverColorMap.

62.1.27.3.11 TRibbonSilverColorMap.FontColor Property

Changes the look and feel of your applications.

Pascal

```
property FontColor;
```

C++

```
__property FontColor;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.27.3.12 TRibbonSilverColorMap.GroupFontColor Property

Pascal

```
property GroupFontColor: TColor;
```

C++

```
__property TColor GroupFontColor;
```

Description

This is GroupFontColor, a member of class TRibbonSilverColorMap.

62.1.27.3.13 TRibbonSilverColorMap.HighlightColor Property

Changes the look and feel of your applications.

Pascal

```
property HighlightColor;
```

C++

```
__property HighlightColor;
```

Description

Use Object Inspector for the respective colormaps to change the default settings and customize the look and feel of your applications.

62.1.27.3.14 TRibbonSilverColorMap.InactiveTabFontColor Property

Pascal

```
property InactiveTabFontColor: TColor;
```

C++

```
__property TColor InactiveTabFontColor;
```

Description

This is InactiveTabFontColor, a member of class TRibbonSilverColorMap.

62.1.27.3.15 TRibbonSilverColorMap.QuickAccessToolbarMoreBorderColor Property

Pascal

```
property QuickAccessToolbarMoreBorderColor: TColor;
```

C++

```
__property TColor QuickAccessToolbarMoreBorderColor;
```

Description

This is QuickAccessToolbarMoreBorderColor, a member of class TRibbonSilverColorMap.

62.1.27.3.16 TRibbonSilverColorMap.QuickAccessToolbarMoreColor Property

Pascal

```
property QuickAccessToolbarMoreColor: TColor;
```

C++

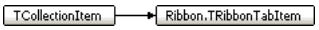
```
__property TColor QuickAccessToolbarMoreColor;
```

Description

This is QuickAccessToolbarMoreColor, a member of class TRibbonSilverColorMap.

62.1.28 TRibbonTabItem Class

Class Hierarchy



Pascal

```
TRibbonTabItem = class(TCollectionItem);
```

C++

```
class TRibbonTabItem : public TCollectionItem;
```

File

Ribbon

Description

This is class Ribbon.TRibbonTabItem.

62.1.28.1 TRibbonTabItem Members

The following tables list the members exposed by TRibbonTabItem.

Protected Methods

	Name	Description
	GetDisplayName (see page 3557)	Returns the name of the collection item as it appears in the collection editor.

Public Methods

	Name	Description
	Assign (see page 3556)	Copies the contents of another, similar object.
	Destroy (see page 3555)	Destroys the TCollectionItem instance and frees its memory.

Public Properties

	Name	Description
	BestTabWidth (see page 3558)	This is BestTabWidth, a member of class TRibbonTabItem.
	KeyTipForm (see page 3558)	This is KeyTipForm, a member of class TRibbonTabItem.
	MinTabwidth (see page 3558)	This is MinTabwidth, a member of class TRibbonTabItem.
	MinTruncatedWidth (see page 3559)	This is MinTruncatedWidth, a member of class TRibbonTabItem.
	Modified (see page 3559)	This is Modified, a member of class TRibbonTabItem.
	X (see page 3559)	This is X, a member of class TRibbonTabItem.
	Y (see page 3560)	This is Y, a member of class TRibbonTabItem.

Published Properties

	Name	Description
	Caption (see page 3558)	This is Caption, a member of class TRibbonTabItem.
	KeyTip (see page 3558)	This is KeyTip, a member of class TRibbonTabItem.
	Page (see page 3559)	This is Page, a member of class TRibbonTabItem.

62.1.28.2 TRibbonTabItem.Destroy Destructor

Destroys the TCollectionItem instance and frees its memory.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRibbonTabItem();
```

Description

Destroy is called indirectly by TCollection's Clear or Delete method.

See Also

Clear

Delete

System::TObject::Free

62.1.28.3 TRibbonTabItem Methods

The methods of the TRibbonTabItem class are listed here.

Protected Methods

	Name	Description
	GetDisplayName (see page 3557)	Returns the name of the collection item as it appears in the collection editor.

Public Methods

	Name	Description
	Assign (see page 3556)	Copies the contents of another, similar object.

62.1.28.3.1 TRibbonTabItem.Assign Method

Copies the contents of another, similar object.

Pascal

```
procedure Assign(Source: TPersistent); override;
```

C++

```
virtual __fastcall Assign(TPersistent * Source);
```

Description

Call Assign to copy the properties or other attributes of one object from another. The standard form of a call to Assign is

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

which tells the Destination object to copy the contents of the Source object to itself.

Most objects override Assign to handle the assignment of properties from similar objects. When overriding Assign, call the inherited method if the destination object can't handle the assignment of properties from the class of the Source parameter.

If no overridden Assign method can handle the assignment of properties from Source, the method implemented in TPersistent calls the source object's AssignTo method. This allows the source object to handle the assignment. If the Source object is nil (Delphi) or NULL (C++), Assign raises an EConvertError exception.

In general, the statement

```
Destination := Source; {Delphi}
Destination = Source; // C++
```

is not the same as the statement

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

The assignment operator makes Destination reference the same object as Source, whereas the Assign method copies the contents of the object referenced by Source into the object referenced by Destination.

Note: The types of some properties are also objects. If these properties have write methods that use Assign to set the value of the property, then in these cases the assignment operator does the same thing as the Assign method.

See Also

AssignTo

EConvertError

62.1.28.3.2 TRibbonTabItem.GetDisplayName Method

Returns the name of the collection item as it appears in the collection editor.

Pascal

```
function GetDisplayName: string; override;
```

C++

```
virtual __fastcall UnicodeString GetDisplayName();
```

Description

GetDisplayName is the protected read implementation of the DisplayName property.

See Also

DisplayName

GetNamePath

62.1.28.4 TRibbonTabItem Properties

The properties of the TRibbonTabItem class are listed here.

Public Properties

	Name	Description
R	BestTabWidth (see page 3558)	This is BestTabWidth, a member of class TRibbonTabItem.
	KeyTipForm (see page 3558)	This is KeyTipForm, a member of class TRibbonTabItem.
R	MinTabwidth (see page 3558)	This is MinTabwidth, a member of class TRibbonTabItem.
R	MinTruncatedWidth (see page 3559)	This is MinTruncatedWidth, a member of class TRibbonTabItem.
	Modified (see page 3559)	This is Modified, a member of class TRibbonTabItem.
	X (see page 3559)	This is X, a member of class TRibbonTabItem.
	Y (see page 3560)	This is Y, a member of class TRibbonTabItem.

Published Properties

	Name	Description
	Caption (see page 3558)	This is Caption, a member of class TRibbonTabItem.
	KeyTip (see page 3558)	This is KeyTip, a member of class TRibbonTabItem.
	Page (see page 3559)	This is Page, a member of class TRibbonTabItem.

62.1.28.4.1 TRibbonTabItem.BestTabWidth Property

Pascal

```
property BestTabWidth: Integer;
```

C++

```
__property int BestTabWidth;
```

Description

This is BestTabWidth, a member of class TRibbonTabItem.

62.1.28.4.2 TRibbonTabItem.Caption Property

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

This is Caption, a member of class TRibbonTabItem.

62.1.28.4.3 TRibbonTabItem.KeyTip Property

Pascal

```
property KeyTip: string;
```

C++

```
__property UnicodeString KeyTip;
```

Description

This is KeyTip, a member of class TRibbonTabItem.

62.1.28.4.4 TRibbonTabItem.KeyTipForm Property

Pascal

```
property KeyTipForm: TCustomControl;
```

C++

```
__property TCustomControl KeyTipForm;
```

Description

This is KeyTipForm, a member of class TRibbonTabItem.

62.1.28.4.5 TRibbonTabItem.MinTabwidth Property

Pascal

```
property MinTabwidth: Integer;
```

C++

```
__property int MinTabwidth;
```

Description

This is MinTabwidth, a member of class TRibbonTabItem.

62.1.28.4.6 TRibbonTabItem.MinTruncatedWidth Property

Pascal

```
property MinTruncatedWidth: Integer;
```

C++

```
__property int MinTruncatedWidth;
```

Description

This is MinTruncatedWidth, a member of class TRibbonTabItem.

62.1.28.4.7 TRibbonTabItem.Modified Property

Pascal

```
property Modified: Boolean;
```

C++

```
__property Boolean Modified;
```

Description

This is Modified, a member of class TRibbonTabItem.

62.1.28.4.8 TRibbonTabItem.Page Property

Pascal

```
property Page: TCustomRibbonPage;
```

C++

```
__property TCustomRibbonPage Page;
```

Description

This is Page, a member of class TRibbonTabItem.

62.1.28.4.9 TRibbonTabItem.X Property

Pascal

```
property X: Integer;
```

C++

```
__property int X;
```

Description

This is X, a member of class TRibbonTabItem.

62.1.28.4.10 TRibbonTabItem.Y Property

Pascal

```
property Y: Integer;
```

C++

```
__property int Y;
```

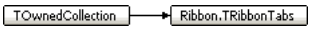
Description

This is Y, a member of class TRibbonTabItem.

62.1.29 TRibbonTabs Class

Collection of information about ribbon pages.

Class Hierarchy



Pascal

```
TRibbonTabs = class(TOwnedCollection);
```

C++

```
class TRibbonTabs : public TOwnedCollection;
```

File

Ribbon

Description

TRibbonTabs is a collection that maintains information about ribbon pages, their captions and keytip values.

The ribbon control can support many user selectable tabs. The TRibbonTabs object stores all tabs and tab specific values associated with the tab. TRibbonTabs also performs the calculations needed to determine if the tabs can fit in the user displayed area and, if not, performs the resizing of the tabs.

62.1.29.1 TRibbonTabs Members

The following tables list the members exposed by TRibbonTabs.

Protected Methods

	Name	Description
	Notify (see page 3562)	Responds when items are added to or removed from the collection.

Public Methods

	Name	Description
	Add (see page 3561)	Creates a new TCollectionItem instance and adds it to the Items array.
	Create (see page 3561)	Creates and initializes a TOwnedCollection instance.
	FindGroup (see page 3562)	This is FindGroup, a member of class TRibbonTabs.
	GetPageFromPageCaption (see page 3562)	This is GetPageFromPageCaption, a member of class TRibbonTabs.
	IndexOfCaption (see page 3562)	This is IndexOfCaption, a member of class TRibbonTabs.
	SelectNext (see page 3563)	Select next page in ribbon.

	SelectPrior (see page 3563)	Select prior page in ribbon.
	TabExists (see page 3564)	This is TabExists, a member of class TRibbonTabs.

Public Properties

	Name	Description
	DesignerAdd (see page 3564)	This is DesignerAdd, a member of class TRibbonTabs.
	Items (see page 3564)	Lists the items in the collection.
R	Ribbon (see page 3565)	This is Ribbon, a member of class TRibbonTabs.

62.1.29.2 TRibbonTabs.Create Constructor

Creates and initializes a TOwnedCollection instance.

Pascal

```
constructor Create(AOwner: TPersistent; ItemClass: TCollectionItemClass);
```

C++

```
__fastcall TRibbonTabs(TPersistent * AOwner, TCollectionItemClass ItemClass);
```

Description

The Create method takes two parameters: the name of an instance object descended from TPersistent and the name of a TCollectionItem descendant class. The first parameter is the owner of the TOwnedCollection instance. The second parameter determines the class of the items created by the Add method.

See Also

Add

62.1.29.3 TRibbonTabs Methods

The methods of the TRibbonTabs class are listed here.

Protected Methods

	Name	Description
V	Notify (see page 3562)	Responds when items are added to or removed from the collection.

Public Methods

	Name	Description
	Add (see page 3561)	Creates a new TCollectionItem instance and adds it to the Items array.
	FindGroup (see page 3562)	This is FindGroup, a member of class TRibbonTabs.
	GetPageFromPageCaption (see page 3562)	This is GetPageFromPageCaption, a member of class TRibbonTabs.
	IndexOfCaption (see page 3562)	This is IndexOfCaption, a member of class TRibbonTabs.
	SelectNext (see page 3563)	Select next page in ribbon.
	SelectPrior (see page 3563)	Select prior page in ribbon.
	TabExists (see page 3564)	This is TabExists, a member of class TRibbonTabs.

62.1.29.3.1 TRibbonTabs.Add Method

Creates a new TCollectionItem instance and adds it to the Items array.

Pascal

```
function Add: TRibbonTabItem;
```

C++

```
__fastcall TRibbonTabItem Add();
```

Description

Call Add to create an item in the collection. The new item is placed at the end of the Items array.

Add returns the new collection item.

See Also

Items

Insert

Create

Delete

62.1.29.3.2 TRibbonTabs.FindGroup Method

Pascal

```
function FindGroup(const GroupName: string): TControl;
```

C++

```
__fastcall TControl * FindGroup(const UnicodeString GroupName);
```

Description

This is FindGroup, a member of class TRibbonTabs.

62.1.29.3.3 TRibbonTabs.GetPageFromPageCaption Method

Pascal

```
function GetPageFromPageCaption(const Caption: string): TCustomRibbonPage;
```

C++

```
__fastcall TCustomRibbonPage GetPageFromPageCaption(const UnicodeString Caption);
```

Description

This is GetPageFromPageCaption, a member of class TRibbonTabs.

62.1.29.3.4 TRibbonTabs.IndexOfCaption Method

Pascal

```
function IndexOfCaption(const Caption: string): Integer;
```

C++

```
__fastcall int IndexOfCaption(const UnicodeString Caption);
```

Description

This is IndexOfCaption, a member of class TRibbonTabs.

62.1.29.3.5 TRibbonTabs.Notify Method

Responds when items are added to or removed from the collection.

Pascal

```
procedure Notify(Item: TCollectionItem; Action: TCollectionNotification); override;
```

C++

```
virtual __fastcall Notify(TCollectionItem Item, TCollectionNotification Action);
```

Description

Notify is called automatically when the items in the collection change.

Item is the item that was just added to or that is about to be removed from the collection.

Action indicates whether item was added, is about to be removed, or is about to be deleted:

As implemented in TCollection, Notify calls Added when `Action` is cnAdded and calls Deleting when `Action` is cnDeleting. TCollection ignores the cnExtracting action. Descendant classes can override Notify to modify this behavior.

See Also

Added

Deleting

62.1.29.3.6 TRibbonTabs.SelectNext Method

Select next page in ribbon.

Pascal

```
function SelectNext: Boolean;
```

C++

```
__fastcall Boolean SelectNext();
```

Description

Use SelectNext to select the next page in the ribbon.

See Also

SelectPrior (see page 3563)

62.1.29.3.7 TRibbonTabs.SelectPrior Method

Select prior page in ribbon.

Pascal

```
function SelectPrior: Boolean;
```

C++

```
__fastcall Boolean SelectPrior();
```

Description

Use SelectPrior to select the prior page in the ribbon.

See Also

SelectNext (see page 3563)

62.1.29.3.8 TRibbonTabs.TabExists Method

Pascal

```
function TabExists(const Caption: string): Boolean;
```

C++

```
__fastcall Boolean TabExists(const UnicodeString Caption);
```

Description

This is TabExists, a member of class TRibbonTabs.

62.1.29.4 TRibbonTabs Properties

The properties of the TRibbonTabs class are listed here.

Public Properties

	Name	Description
	DesignerAdd (see page 3564)	This is DesignerAdd, a member of class TRibbonTabs.
	Items (see page 3564)	Lists the items in the collection.
R	Ribbon (see page 3565)	This is Ribbon, a member of class TRibbonTabs.

62.1.29.4.1 TRibbonTabs.DesignerAdd Property

Pascal

```
property DesignerAdd: Boolean;
```

C++

```
__property Boolean DesignerAdd;
```

Description

This is DesignerAdd, a member of class TRibbonTabs.

62.1.29.4.2 TRibbonTabs.Items Property

Lists the items in the collection.

Pascal

```
property Items [Index: Integer]: TRibbonTabItem;
```

C++

```
__property TRibbonTabItem Items[int Index];
```

Description

Use Items to access individual items in the collection. The value of the `Index` parameter corresponds to the `Index` property of TCollectionItem. It represents the position of the item in the collection.

See Also

Count

ItemClass

Index

Add

Insert

62.1.29.4.3 TRibbonTabs.Ribbon Property

Pascal

```
property Ribbon: TCustomRibbon;
```

C++

```
__property TCustomRibbon Ribbon;
```

Description

This is Ribbon, a member of class TRibbonTabs.

62.2 Interfaces

The following table lists interfaces in this documentation.

Interfaces

	Name	Description
	IRibbon (see page 3566)	Interface for ribbon control.
	IRibbonGroup (see page 3568)	Interface for any ribbon-specific group.
	IRibbonQuickAccessToolbar (see page 3569)	This is class Ribbon.IRibbonQuickAccessToolbar.

62.2.1 IRibbon Interface

Interface for ribbon control.

Class Hierarchy

Ribbon.IRibbon

Pascal

```
IRibbon = interface;
```

C++

```
__interface IRibbon;
```

File

Ribbon

Description

Implement the IRibbon interface to acknowledge that the control supports the ribbon control.

See Also

Creating an Application that Uses Ribbon Controls

Adding Commands to the Ribbon

IRibbonGroup (see page 3568)

62.2.1.1 IRibbon Members

The following tables list the members exposed by IRibbon.

Public Methods

	Name	Description
	GetRibbon (see page 3567)	This is GetRibbon, a member of class IRibbon.
	SetRibbon (see page 3567)	This is SetRibbon, a member of class IRibbon.

Public Properties

	Name	Description
	Ribbon (see page 3567)	This is Ribbon, a member of class IRibbon.

62.2.1.2 IRibbon Methods

The methods of the IRibbon class are listed here.

Public Methods

	Name	Description
	GetRibbon (see page 3567)	This is GetRibbon, a member of class IRibbon.
	SetRibbon (see page 3567)	This is SetRibbon, a member of class IRibbon.

62.2.1.2.1 IRibbon.GetRibbon Method

Pascal

```
function GetRibbon: TCustomRibbon;
```

C++

```
__fastcall TCustomRibbon GetRibbon();
```

Description

This is GetRibbon, a member of class IRibbon.

62.2.1.2.2 IRibbon.SetRibbon Method

Pascal

```
procedure SetRibbon(const Value: TCustomRibbon);
```

C++

```
__fastcall SetRibbon(const TCustomRibbon Value);
```

Description

This is SetRibbon, a member of class IRibbon.

62.2.1.3 IRibbon Properties

The properties of the IRibbon class are listed here.

Public Properties

	Name	Description
	Ribbon (see page 3567)	This is Ribbon, a member of class IRibbon.

62.2.1.3.1 IRibbon.Ribbon Property

Pascal

```
property Ribbon: TCustomRibbon;
```

C++

```
__property TCustomRibbon Ribbon;
```

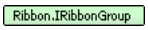
Description

This is Ribbon, a member of class IRibbon.

62.2.2 IRibbonGroup Interface

Interface for any ribbon-specific group.

Class Hierarchy



Pascal

```
IRibbonGroup = interface;
```

C++

```
__interface IRibbonGroup;
```

File

Ribbon

Description

IRibbonGroup is an interface implemented by any ribbon-specific group.

See Also

IRibbon (see page 3566)

62.2.2.1 IRibbonGroup Members

The following tables list the members exposed by IRibbonGroup.

Public Methods

	Name	Description
	AlignGroupControls (see page 3568)	Realign controls on group.
	CreateControl (see page 3569)	Recreate specific control.
	StyleChanged (see page 3569)	This is StyleChanged, a member of class IRibbonGroup.

62.2.2.2 IRibbonGroup Methods

The methods of the IRibbonGroup class are listed here.

Public Methods

	Name	Description
	AlignGroupControls (see page 3568)	Realign controls on group.
	CreateControl (see page 3569)	Recreate specific control.
	StyleChanged (see page 3569)	This is StyleChanged, a member of class IRibbonGroup.

62.2.2.2.1 IRibbonGroup.AlignGroupControls Method

Realign controls on group.

Pascal

```
procedure AlignGroupControls;
```

C++

```
__fastcall AlignGroupControls();
```

Description

AlignGroupControls realigns controls on a group.

See Also

CreateControl (see page 3569)

62.2.2.2.2 IRibbonGroup.CreateControl Method

Recreate specific control.

Pascal

```
function CreateControl(AnItem: TActionClientItem): TCustomActionControl;
```

C++

```
__fastcall TCustomActionControl CreateControl(TActionClientItem AnItem);
```

Description

CreateControl recreates a specific control.

See Also

AlignGroupControls (see page 3568)

62.2.2.2.3 IRibbonGroup.StyleChanged Method

Pascal

```
procedure StyleChanged(AStyle: TRibbonStyleActionBars);
```

C++

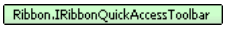
```
__fastcall StyleChanged(TRibbonStyleActionBars AStyle);
```

Description

This is StyleChanged, a member of class IRibbonGroup.

62.2.3 IRibbonQuickAccessToolbar Interface

Class Hierarchy



Pascal

```
IRibbonQuickAccessToolbar = interface;
```

C++

```
__interface IRibbonQuickAccessToolbar;
```

File

Ribbon

Description

This is class Ribbon.IRibbonQuickAccessToolbar.

62.2.3.1 IRibbonQuickAccessToolbar Members

The following tables list the members exposed by IRibbonQuickAccessToolbar.

Public Methods

	Name	Description
	DoUpdateQATWidth (see page 3570)	This is DoUpdateQATWidth, a member of class IRibbonQuickAccessToolbar.

62.2.3.2 IRibbonQuickAccessToolbar Methods

The methods of the IRibbonQuickAccessToolbar class are listed here.

Public Methods

	Name	Description
	DoUpdateQATWidth (see page 3570)	This is DoUpdateQATWidth, a member of class IRibbonQuickAccessToolbar.

62.2.3.2.1 IRibbonQuickAccessToolbar.DoUpdateQATWidth Method

Pascal

```
procedure DoUpdateQATWidth;
```

C++

```
__fastcall DoUpdateQATWidth();
```

Description

This is DoUpdateQATWidth, a member of class IRibbonQuickAccessToolbar.

62.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TCommandType (see page 3571)	TCommandType enumerates the possible types of items to be displayed in the right pane of the Application Menu.
	TIconSize (see page 3572)	This is record Ribbon.TIconSize.
	TRibbonGroupVariant (see page 3572)	This is record Ribbon.TRibbonGroupVariant.
	TRibbonMetric (see page 3573)	This is record Ribbon.TRibbonMetric.
	TRibbonScrollButton (see page 3573)	This is record Ribbon.TRibbonScrollButton.
	TTabActiveScrollButton (see page 3574)	This is record Ribbon.TTabActiveScrollButton.
	TTabSeparatorState (see page 3574)	Position of Quick Access Toolbar.

62.3.1 Ribbon.TCommandType Enumeration

TCommandType enumerates the possible types of items to be displayed in the right pane of the Application Menu.

Pascal

```
TCommandType = (
  ctRecent,
  ctCommands
);
```

C++

```
enum TCommandType {
  ctRecent,
  ctCommands
};
```

File

Ribbon

Description

TCommandType represents the types of items that will be displayed in the right pane of the Application Menu. The possible values are.

Value	Description
ctRecent	A list of recently used items will be displayed in the right pane of the application menu.
ctCommands	A list of commands will be displayed in the right pane of the application menu.

Note: Recent items are automatically prefixed with a number accelerator key for quick use.

See Also

TIconSize (see page 3572)

TApplicationMenu (see page 3350)

CommandType (see page 3354)

62.3.2 Ribbon.TIconSize Enumeration

Pascal

```
TIconSize = (
  isSmall,
  isMedium,
  isLarge
);
```

C++

```
enum TIconSize {
  isSmall,
  isMedium,
  isLarge
};
```

File

Ribbon

Description

This is record Ribbon.TIconSize.

62.3.3 Ribbon.TRibbonGroupVariant Enumeration

Pascal

```
TRibbonGroupVariant = (
  rgvCollapsed,
  rgvSmall,
  rgvMedium,
  rgvLarge
);
```

C++

```
enum TRibbonGroupVariant {
  rgvCollapsed,
  rgvSmall,
  rgvMedium,
  rgvLarge
};
```

File

Ribbon

Description

This is record Ribbon.TRibbonGroupVariant.

62.3.4 Ribbon.TRibbonMetric Enumeration

Pascal

```
  TRibbonMetric = (
    rmCaption,
    rmHeight,
    rmFullHeight,
    rmMinHeight,
    rmTabTop,
    rmPageHeight,
    rmGroupHeight,
    rmGroupCaptionHeight,
    rmQATWidth,
    rmSystemButtonHeight,
    rmSystemButtonWidth,
    rmTabHeight,
    rmFontSize
  );
```

C++

```
  enum TRibbonMetric {
    rmCaption,
    rmHeight,
    rmFullHeight,
    rmMinHeight,
    rmTabTop,
    rmPageHeight,
    rmGroupHeight,
    rmGroupCaptionHeight,
    rmQATWidth,
    rmSystemButtonHeight,
    rmSystemButtonWidth,
    rmTabHeight,
    rmFontSize
  };
```

File

Ribbon

Description

This is record Ribbon.TRibbonMetric.

62.3.5 Ribbon.TRibbonScrollButton Enumeration

Pascal

```
  TRibbonScrollButton = (
    tsbTabRight,
    tsbTabLeft,
    tsbGroupLeft,
    tsbGroupRight
  );
```

C++

```
  enum TRibbonScrollButton {
```

```
    tsbTabRight,
    tsbTabLeft,
    tsbGroupLeft,
    tsbGroupRight
  };
```

File

Ribbon

Description

This is record Ribbon.TRibbonScrollButton.

62.3.6 Ribbon.TTabActiveScrollButton Enumeration

Pascal

```
  TTabActiveScrollButton = (
    asbNone,
    asbRight,
    asbLeft
  );
```

C++

```
  enum TTabActiveScrollButton {
    asbNone,
    asbRight,
    asbLeft
  };
```

File

Ribbon

Description

This is record Ribbon.TTabActiveScrollButton.

62.3.7 Ribbon.TTabSeparatorState Enumeration

Position of Quick Access Toolbar.

Pascal

```
  TTabSeparatorState = (
    tssHidden,
    tssLight,
    tssMedium,
    tssDark
  );
```

C++

```
  enum TTabSeparatorState {
    tssHidden,
    tssLight,
    tssMedium,
    tssDark
  };
```

File

Ribbon

Description

TTabSeparatorState is the state of the separators that are displayed between tabs when the ribbon is being resized smaller.

Value	Meaning
tssHidden	Hidden
tssLight	Light
tssMedium	Medium
tssDark	Dark

See Also

TApplicationMenu (see page 3350)

62.4 Types

The following table lists types in this documentation.

Types

Name	Description
TCustomRibbonClass (see page 3576)	This is type Ribbon.TCustomRibbonClass.
TRecentItemClickEvent (see page 3576)	This is type Ribbon.TRecentItemClickEvent.
TRibbonScrollButtons (see page 3577)	This is type Ribbon.TRibbonScrollButtons.

62.4.1 Ribbon.TCustomRibbonClass Type

Pascal

```
TCustomRibbonClass = class of TCustomRibbon;
```

C++

```
TCustomRibbon TCustomRibbonClass;
```

File

Ribbon

Description

This is type Ribbon.TCustomRibbonClass.

62.4.2 Ribbon.TRecentItemClickEvent Type

Pascal

```
TRecentItemClickEvent = procedure (Sender: TObject; FileName: string; Index: Integer) of
object;
```

C++

```
(Sender: TObject; FileName: string; Index: Integer) ( TRecentItemClickEvent)();
```

File

Ribbon

Description

This is type Ribbon.TRecentItemClickEvent.

62.4.3 Ribbon.TRibbonScrollButtons Type

Pascal

```
TRibbonScrollButtons = set of TRibbonScrollButton;
```

C++

```
TRibbonScrollButton TRibbonScrollButtons;
```

File

Ribbon

Description

This is type Ribbon.TRibbonScrollButtons.

63 RibbonActnCtrls Namespace

This is namespace RibbonActnCtrls.

Topics

Name	Description
Classes (see page 3582)	The following table lists classes in this documentation.
Interfaces (see page 3912)	The following table lists interfaces in this documentation.
Functions (see page 3924)	The following table lists functions in this documentation.
Structs, Records, Enums (see page 3926)	The following table lists structs, records, enums in this documentation.
Types (see page 3931)	The following table lists types in this documentation.
Constants (see page 3932)	The following table lists constants in this documentation.

Classes

	Name	Description
	TCustomRibbonComboBox (see page 3583)	This is class RibbonActnCtrls.TCustomRibbonComboBox.
	TKeyTipList (see page 3592)	**KeyTip** list.
	TKeyTipListEnumerator (see page 3596)	This is class RibbonActnCtrls.TKeyTipListEnumerator.
	TKeyTipMatchItem (see page 3598)	Details about Full and Partial **KeyTip** matches.
	TKeyTipValidator (see page 3600)	Used to validate **KeyTip**.
	TRibbonAddRemoveItem (see page 3601)	This is class RibbonActnCtrls.TRibbonAddRemoveItem.
	TRibbonApplicationButton (see page 3603)	This is class RibbonActnCtrls.TRibbonApplicationButton.
	TRibbonApplicationMenuButton (see page 3604)	Application Button when using Ribbon (see page 3347) interface.
	TRibbonApplicationMenuItem (see page 3612)	This is class RibbonActnCtrls.TRibbonApplicationMenuItem.
	TRibbonBaseButtonControl (see page 3616)	Common ancestor for many buttons in groups and Quick Access Toolbar.
	TRibbonCheckBoxControl (see page 3626)	This is class RibbonActnCtrls.TRibbonCheckBoxControl.
	TRibbonComboBox (see page 3628)	This is class RibbonActnCtrls.TRibbonComboBox.
	TRibbonComboBoxItem (see page 3682)	This is class RibbonActnCtrls.TRibbonComboBoxItem.
	TRibbonComboControl (see page 3684)	This is class RibbonActnCtrls.TRibbonComboControl.
	TRibbonCustomControl (see page 3698)	This is class RibbonActnCtrls.TRibbonCustomControl.
	TRibbonCustomizeItem (see page 3710)	This is class RibbonActnCtrls.TRibbonCustomizeItem.
	TRibbonCustomStatusControl (see page 3713)	This is class RibbonActnCtrls.TRibbonCustomStatusControl.
	TRibbonDropDownButton (see page 3715)	This is class RibbonActnCtrls.TRibbonDropDownButton.
	TRibbonGallery (see page 3720)	This is class RibbonActnCtrls.TRibbonGallery.
	TRibbonGalleryControl (see page 3720)	This is class RibbonActnCtrls.TRibbonGalleryControl.
	TRibbonGalleryControlMenuItem (see page 3722)	This is class RibbonActnCtrls.TRibbonGalleryControlMenuItem.
	TRibbonGalleryItem (see page 3728)	This is class RibbonActnCtrls.TRibbonGalleryItem.

	TRibbonGalleryMenuItem (see page 3730)	This is class RibbonActnCtrls.TRibbonGalleryMenuItem.
	TRibbonGallerySplitButton (see page 3733)	This is class RibbonActnCtrls.TRibbonGallerySplitButton.
	TRibbonGripItem (see page 3735)	This is class RibbonActnCtrls.TRibbonGripItem.
	TRibbonGroupDropDown (see page 3738)	Display entire group as dropdown menu in Quick Access Toolbar.
	TRibbonLargeButtonControl (see page 3742)	This is class RibbonActnCtrls.TRibbonLargeButtonControl.
	TRibbonLargeSplitButton (see page 3745)	This is class RibbonActnCtrls.TRibbonLargeSplitButton.
	TRibbonMainMenuItem (see page 3752)	This is class RibbonActnCtrls.TRibbonMainMenuItem.
	TRibbonMainMenuSeparator (see page 3754)	This is class RibbonActnCtrls.TRibbonMainMenuSeparator.
	TRibbonMenuItem (see page 3755)	This is class RibbonActnCtrls.TRibbonMenuItem.
	TRibbonMenuSeparatorItem (see page 3760)	This is class RibbonActnCtrls.TRibbonMenuSeparatorItem.
	TRibbonMenuSmallButtonControl (see page 3762)	This is class RibbonActnCtrls.TRibbonMenuSmallButtonControl.
	TRibbonOptionButtonControl (see page 3765)	This is class RibbonActnCtrls.TRibbonOptionButtonControl.
	TRibbonQATScrollButton (see page 3767)	This is class RibbonActnCtrls.TRibbonQATScrollButton.
	TRibbonRadioButtonControl (see page 3771)	This is class RibbonActnCtrls.TRibbonRadioButtonControl.
	TRibbonRecentButtonControl (see page 3774)	This is class RibbonActnCtrls.TRibbonRecentButtonControl.
	TRibbonRichContentItem (see page 3776)	This is class RibbonActnCtrls.TRibbonRichContentItem.
	TRibbonRichContentMainMenuItem (see page 3780)	This is class RibbonActnCtrls.TRibbonRichContentMainMenuItem.
	TRibbonSeparator (see page 3784)	This is class RibbonActnCtrls.TRibbonSeparator.
	TRibbonSmallButtonControl (see page 3787)	This is class RibbonActnCtrls.TRibbonSmallButtonControl.
	TRibbonSmallSplitButton (see page 3789)	This is class RibbonActnCtrls.TRibbonSmallSplitButton.
	TRibbonSpinButton (see page 3796)	This is class RibbonActnCtrls.TRibbonSpinButton.
	TRibbonSpinEdit (see page 3836)	This is class RibbonActnCtrls.TRibbonSpinEdit.
	TRibbonSplitMenuItem (see page 3896)	This is class RibbonActnCtrls.TRibbonSplitMenuItem.
	TRibbonStyleMenuButton (see page 3902)	Top level menus when using Ribbon (see page 3347) Styles in applications not using Ribbon (see page 3347) interface.
	TRibbonTextControl (see page 3903)	This is class RibbonActnCtrls.TRibbonTextControl.
	TRibbonToolScrollButton (see page 3905)	This is class RibbonActnCtrls.TRibbonToolScrollButton.
	TTimerSpeedButton (see page 3906)	This is class RibbonActnCtrls.TTimerSpeedButton.

Constants

Name	Description
UM_FADECONTROL (see page 3932)	This is constant RibbonActnCtrls.UM_FADECONTROL.

Functions

	Name	Description
	DrawLine (see page 3924)	This is function RibbonActnCtrls.DrawLine.
	IsLeftMouseButtonPressed (see page 3924)	This is function RibbonActnCtrls.IsLeftMouseButtonPressed.
	ShowKeyTip (see page 3924)	Show **KeyTip** for control.
	ShowKeyTip (see page 3924)	Show **KeyTip** for control.

Interfaces

	Name	Description
	IRibbonBarKeyTip (see page 3912)	Action Bars that support KeyTips should implement this interface.
	IRibbonContainedControl (see page 3914)	Interface implemented by controls used as containers for displaying other controls.
	IRibbonControl (see page 3916)	Interface implemented by controls visible in Ribbon (see page 3347) Groups.
	IRibbonGroupDropdown (see page 3917)	This is class RibbonActnCtrls.IRibbonGroupDropdown.
	IRibbonKeyTip (see page 3918)	Commands with a **KeyTip (see page 3922)** should implement this interface.
	IRibbonSplitButton (see page 3922)	Interface implemented by controls that have split button.

Structs, Records, Enums

	Name	Description
	TKeyTipHandled (see page 3926)	Validate if command's **KeyTip** matches current user input.
	TKeyTipMatchType (see page 3927)	Type of match command's **KeyTip** is for current **KeyTip** value.
	TKeyTipPosition (see page 3927)	Position of KeyTips.

	TKeyTipState (see page 3928)	Current state of KeyTips.
	TQATScrollMode (see page 3929)	This is record RibbonActnCtrls.TQATScrollMode.
	TSpinButtonStyle (see page 3930)	This is record RibbonActnCtrls.TSpinButtonStyle.

Types

Name	Description
TNumGlyphs (see page 3931)	This is type RibbonActnCtrls.TNumGlyphs.
TTimeBtnState (see page 3931)	This is type RibbonActnCtrls.TTimeBtnState.

63.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TCustomRibbonComboBox (see page 3583)	This is class RibbonActnCtrls.TCustomRibbonComboBox.
	TKeyTipList (see page 3592)	**KeyTip** list.
	TKeyTipListEnumerator (see page 3596)	This is class RibbonActnCtrls.TKeyTipListEnumerator.
	TKeyTipMatchItem (see page 3598)	Details about Full and Partial **KeyTip** matches.
	TKeyTipValidator (see page 3600)	Used to validate **KeyTip**.
	TRibbonAddRemoveItem (see page 3601)	This is class RibbonActnCtrls.TRibbonAddRemoveItem.
	TRibbonApplicationButton (see page 3603)	This is class RibbonActnCtrls.TRibbonApplicationButton.
	TRibbonApplicationMenuButton (see page 3604)	Application Button when using Ribbon (see page 3347) interface.
	TRibbonApplicationMenuItem (see page 3612)	This is class RibbonActnCtrls.TRibbonApplicationMenuItem.
	TRibbonBaseButtonControl (see page 3616)	Common ancestor for many buttons in groups and Quick Access Toolbar.
	TRibbonCheckBoxControl (see page 3626)	This is class RibbonActnCtrls.TRibbonCheckBoxControl.
	TRibbonComboBox (see page 3628)	This is class RibbonActnCtrls.TRibbonComboBox.
	TRibbonComboBoxItem (see page 3682)	This is class RibbonActnCtrls.TRibbonComboBoxItem.
	TRibbonComboControl (see page 3684)	This is class RibbonActnCtrls.TRibbonComboControl.
	TRibbonCustomControl (see page 3698)	This is class RibbonActnCtrls.TRibbonCustomControl.
	TRibbonCustomizeItem (see page 3710)	This is class RibbonActnCtrls.TRibbonCustomizeItem.
	TRibbonCustomStatusControl (see page 3713)	This is class RibbonActnCtrls.TRibbonCustomStatusControl.
	TRibbonDropDownButton (see page 3715)	This is class RibbonActnCtrls.TRibbonDropDownButton.
	TRibbonGallery (see page 3720)	This is class RibbonActnCtrls.TRibbonGallery.
	TRibbonGalleryControl (see page 3720)	This is class RibbonActnCtrls.TRibbonGalleryControl.
	TRibbonGalleryControlMenuItem (see page 3722)	This is class RibbonActnCtrls.TRibbonGalleryControlMenuItem.
	TRibbonGalleryItem (see page 3728)	This is class RibbonActnCtrls.TRibbonGalleryItem.
	TRibbonGalleryMenuItem (see page 3730)	This is class RibbonActnCtrls.TRibbonGalleryMenuItem.
	TRibbonGallerySplitButton (see page 3733)	This is class RibbonActnCtrls.TRibbonGallerySplitButton.
	TRibbonGripItem (see page 3735)	This is class RibbonActnCtrls.TRibbonGripItem.
	TRibbonGroupDropDown (see page 3738)	Display entire group as dropdown menu in Quick Access Toolbar.
	TRibbonLargeButtonControl (see page 3742)	This is class RibbonActnCtrls.TRibbonLargeButtonControl.
	TRibbonLargeSplitButton (see page 3745)	This is class RibbonActnCtrls.TRibbonLargeSplitButton.
	TRibbonMainMenuItem (see page 3752)	This is class RibbonActnCtrls.TRibbonMainMenuItem.
	TRibbonMainMenuSeparator (see page 3754)	This is class RibbonActnCtrls.TRibbonMainMenuSeparator.
	TRibbonMenuItem (see page 3755)	This is class RibbonActnCtrls.TRibbonMenuItem.
	TRibbonMenuSeparatorItem (see page 3760)	This is class RibbonActnCtrls.TRibbonMenuSeparatorItem.
	TRibbonMenuSmallButtonControl (see page 3762)	This is class RibbonActnCtrls.TRibbonMenuSmallButtonControl.
	TRibbonOptionButtonControl (see page 3765)	This is class RibbonActnCtrls.TRibbonOptionButtonControl.
	TRibbonQATScrollButton (see page 3767)	This is class RibbonActnCtrls.TRibbonQATScrollButton.
	TRibbonRadioButtonControl (see page 3771)	This is class RibbonActnCtrls.TRibbonRadioButtonControl.
	TRibbonRecentButtonControl (see page 3774)	This is class RibbonActnCtrls.TRibbonRecentButtonControl.
	TRibbonRichContentItem (see page 3776)	This is class RibbonActnCtrls.TRibbonRichContentItem.
	TRibbonRichContentMainMenuItem (see page 3780)	This is class RibbonActnCtrls.TRibbonRichContentMainMenuItem.
	TRibbonSeparator (see page 3784)	This is class RibbonActnCtrls.TRibbonSeparator.
	TRibbonSmallButtonControl (see page 3787)	This is class RibbonActnCtrls.TRibbonSmallButtonControl.
	TRibbonSmallSplitButton (see page 3789)	This is class RibbonActnCtrls.TRibbonSmallSplitButton.
	TRibbonSpinButton (see page 3796)	This is class RibbonActnCtrls.TRibbonSpinButton.
	TRibbonSpinEdit (see page 3836)	This is class RibbonActnCtrls.TRibbonSpinEdit.

	TRibbonSplitMenuItem (see page 3896)	This is class RibbonActnCtrls.TRibbonSplitMenuItem.
	TRibbonStyleMenuButton (see page 3902)	Top level menus when using Ribbon (see page 3347) Styles in applications not using Ribbon (see page 3347) interface.
	TRibbonTextControl (see page 3903)	This is class RibbonActnCtrls.TRibbonTextControl.
	TRibbonToolScrollButton (see page 3905)	This is class RibbonActnCtrls.TRibbonToolScrollButton.
	TTimerSpeedButton (see page 3906)	This is class RibbonActnCtrls.TTimerSpeedButton.

63.1.1 TCustomRibbonComboBox Class

Class Hierarchy

Pascal

```
TCustomRibbonComboBox = class(TCustomEdit);
```

C++

```
class TCustomRibbonComboBox : public TCustomEdit;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TCustomRibbonComboBox.

63.1.1.1 TCustomRibbonComboBox Members

The following tables list the members exposed by TCustomRibbonComboBox.

Protected Methods

	Name	Description
	BeginAutoDrag (see page 3585)	Begin Automatic Drag.
	CMMouseenter (see page 3585)	This is CMMouseenter, a member of class TCustomRibbonComboBox.
	CMMouseleave (see page 3586)	This is CMMouseleave, a member of class TCustomRibbonComboBox.
	DragOver (see page 3586)	OnDragOver event dispatcher.
	KeyDown (see page 3587)	Respond to key press events.
	KeyPress (see page 3587)	Respond to keyboard input.
	MouseMove (see page 3588)	Respond to mouse moving over control area..
	Notification (see page 3588)	Responds to notifications that components are being created or destroyed.
	SetParent (see page 3589)	Sets the parent of the control.
	WMKillFocus (see page 3590)	This is WMKillFocus, a member of class TCustomRibbonComboBox.
	WMSetFocus (see page 3590)	This is WMSetFocus, a member of class TCustomRibbonComboBox.

Public Methods

	Name	Description
	Create (see page 3584)	Creates an instance of TCustomEdit.
	Destroy (see page 3584)	Destroys an instance of TWinControl.
	DragDrop (see page 3586)	OnDragDrop event dispatcher.
	SetBounds (see page 3589)	Sets the windowed control's boundary properties all at once.

Public Properties

	Name	Description
	Color (see page 3591)	Specifies the background color of the control.
	Items (see page 3592)	This is Items, a member of class TCustomRibbonComboBox.
	MouseInControl (see page 3592)	This is MouseInControl, a member of class TCustomRibbonComboBox.

63.1.1.2 TCustomRibbonComboBox.Create Constructor

Creates an instance of TCustomEdit.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomRibbonComboBox(TComponent * AOwner);
```

Description

Call Create to create and initialize an instance of TCustomEdit.

AOwner is another component, typically the form, that is responsible for freeing the edit control. It becomes the value of the Owner property.

Derived objects that override the Create method should call the inherited Create before performing any additional initialization.

See Also

Owner

63.1.1.3 TCustomRibbonComboBox.Destroy Destructor

Destroys an instance of TWinControl.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomRibbonComboBox();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TWinControl reference is not nil, and only then calls Destroy.

See Also

System::TObject::Free

Destroy

63.1.1.4 TCustomRibbonComboBox Methods

The methods of the TCustomRibbonComboBox class are listed here.

Protected Methods

	Name	Description
	BeginAutoDrag (see page 3585)	Begin Automatic Drag.
	CMMouseenter (see page 3585)	This is CMMouseenter, a member of class TCustomRibbonComboBox.
	CMMouseleave (see page 3586)	This is CMMouseleave, a member of class TCustomRibbonComboBox.
	DragOver (see page 3586)	OnDragOver event dispatcher.
	KeyDown (see page 3587)	Respond to key press events.
	KeyPress (see page 3587)	Respond to keyboard input.
	MouseMove (see page 3588)	Respond to mouse moving over control area..
	Notification (see page 3588)	Responds to notifications that components are being created or destroyed.
	SetParent (see page 3589)	Sets the parent of the control.
	WMKillFocus (see page 3590)	This is WMKillFocus, a member of class TCustomRibbonComboBox.
	WMSetFocus (see page 3590)	This is WMSetFocus, a member of class TCustomRibbonComboBox.

Public Methods

	Name	Description
	DragDrop (see page 3586)	OnDragDrop event dispatcher.
	SetBounds (see page 3589)	Sets the windowed control's boundary properties all at once.

63.1.1.4.1 TCustomRibbonComboBox.BeginAutoDrag Method

Begin Automatic Drag.

Pascal

```
procedure BeginAutoDrag; override;
```

C++

```
virtual __fastcall BeginAutoDrag();
```

Description

BeginAutoDrag is called automatically when a user drags a control whose DragMode is dmAutomatic. Do not call BeginAutoDrag in application code.

As implemented in TControl, BeginAutoDrag calls BeginDrag, passing in the global Mouse variable's DragImmediate and DragThreshold properties.

See Also

DragKind

DragMode

EndDrag

Mouse

DragImmediate

DragThreshold

63.1.1.4.2 TCustomRibbonComboBox.CMMouseenter Method

Pascal

```
procedure CMMouseenter(var Message: TMessage); message CM_MOUSEENTER;
```

C++

```
__fastcall CMMouseenter(TMessage Message);
```

Description

This is CMMouseenter, a member of class TCustomRibbonComboBox.

63.1.1.4.3 TCustomRibbonComboBox.CMMouseleave Method

Pascal

```
procedure CMMouseleave(var Message: TMessage); message CM_MOUSELEAVE;
```

C++

```
__fastcall CMMouseleave(TMessage Message);
```

Description

This is CMMouseleave, a member of class TCustomRibbonComboBox.

63.1.1.4.4 TCustomRibbonComboBox.DragDrop Method

OnDragDrop event dispatcher.

Pascal

```
procedure DragDrop(Source: TObject; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall DragDrop(TObject * Source, int X, int Y);
```

Description

When creating a component that descends from TControl, override DragDrop to add additional code that executes before the OnDragDrop event handler is called.

The Source parameter is the object that was dropped onto the control. The X and Y parameters are the mouse coordinates where the object was dropped.

See Also

DragCanceled

OnDragDrop

63.1.1.4.5 TCustomRibbonComboBox.DragOver Method

OnDragOver event dispatcher.

Pascal

```
procedure DragOver(Source: TObject; X: Integer; Y: Integer; State: TDragState; var Accept:
Boolean); override;
```

C++

```
virtual __fastcall DragOver(TObject * Source, int X, int Y, TDragState State, Boolean Accept);
```

Description

Override DragOver to add additional code that executes before the OnDragOver event handler is called.

DragOver sets the Accept parameter to true to indicate that the user can drop the dragged object on the control. It sets Accept to

false to indicate that the user cannot drop the dragged object on the control.

The Source parameter is the object being dragged. The State parameter indicates how the dragged object is moving in relation to the control. X and Y indicate the current position of the mouse.

See Also

DragDrop

OnDragOver

63.1.1.4.6 TCustomRibbonComboBox.KeyDown Method

Respond to key press events.

Pascal

```
procedure KeyDown(var Key: Word; Shift: TShiftState); override;
```

C++

```
virtual __fastcall KeyDown(Word Key, TShiftState Shift);
```

Description

When a windowed control receives a key-down message (WM_KEYDOWN) from Windows, its message handler calls the DoKeyDown method. If DoKeyDown determines that the control should, in fact, process the character, it calls KeyDown, passing the key code and shift-key state in the Key and Shift parameters, respectively.

KeyDown calls any event handler attached to the OnKeyDown event. Override KeyDown to provide other responses in addition to the event-handler call.

The Key parameter is the key on the keyboard. For non-alphanumeric keys, you must use WinAPI virtual key codes to determine the key pressed. For more information, search for virtual key codes in the Win32 Developer's Reference (WIN32.HLP).

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

Either KeyDown or the OnKeyDown event handler it calls can suppress further processing of a key by setting the Key parameter to zero.

See Also

KeyUp

OnKeyDown

DoKeyDown

DoKeyPress

DoKeyUp

63.1.1.4.7 TCustomRibbonComboBox.KeyPress Method

Respond to keyboard input.

Pascal

```
procedure KeyPress(var Key: Char); override;
```

C++

```
virtual __fastcall KeyPress(Char Key);
```

Description

When a windowed control receives a key-press message (WM_CHAR) from Windows, its message handler calls the DoKeyPress method. If DoKeyPress determines that the control should, in fact, process the character, it calls KeyPress, passing the key code in the Key parameter.

KeyPress calls any event handler attached to the OnKeyPress event. Override KeyPress to provide other responses in addition to the event-handler call.

Either KeyPress or the OnKeyPress event handler it calls can suppress further processing of a character by setting the Key parameter to zero.

Note: The Key parameter is the character represented by the key that is pressed, not a Windows virtual key code.

See Also

KeyDown

KeyUp

OnKeyPress

63.1.1.4.8 TCustomRibbonComboBox.MouseMove Method

Respond to mouse moving over control area..

Pascal

```
procedure MouseMove(Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseMove(TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseMove method to provide other responses in addition to calling the OnMouseMove event handler when the user moves the mouse.

A control calls MouseMove in response to any of the Windows mouse-move messages (WM_MOUSEMOVE), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively.

As the mouse cursor moves across a control, this method is called repeatedly. Each time it is called, it is with the new coordinates that reflect the continuous path of the mouse cursor across the screen real estate covered by the control's visual representation.

See Also

MouseDown

MouseUp

OnMouseMove

63.1.1.4.9 TCustomRibbonComboBox.Notification Method

Responds to notifications that components are being created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification allows TControl to update its PopupMenu property when the pop-up menu it refers to is destroyed.

Override the Notification method to respond to notifications that other components are about to be destroyed or have just been created. Use the Notification method to update controls that rely on other objects. Remove references to objects that are going away so that the control does not refer to objects that have been destroyed. For example, data-aware objects override the Notification method to update themselves when their data source is removed. Some objects respond to notifications that other objects of a specific type are being created. For example, the session component sets the Session property of new data-aware controls to itself when the AutoSessionName property is true.

By default, components pass along the notification to their owned components, if any.

See Also

PopupMenu

Notification

63.1.1.4.10 TCustomRibbonComboBox.SetBounds Method

Sets the windowed control's boundary properties all at once.

Pascal

```
procedure SetBounds(ALeft: Integer; ATop: Integer; AWidth: Integer; AHeight: Integer);
override;
```

C++

```
virtual __fastcall SetBounds(int ALeft, int ATop, int AWidth, int AHeight);
```

Description

Use SetBounds to change all of the control's boundary properties at once. The same effect can be achieved by setting the Left, Top, Width, and Height properties separately. By setting all four properties at once, SetBounds ensures that the control will not repaint between changes

Specify the values for the Left, Top, Width, and Height properties as the value of the ALeft, ATop, AWidth, and AHeight parameters, respectively.

See Also

Height

Left

Top

Width

63.1.1.4.11 TCustomRibbonComboBox.SetParent Method

Sets the parent of the control.

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

SetParent is the protected implementation of the Parent property. Override SetParent to execute additional code when the value of the Parent property changes.

The AParent parameter specifies the new parent of the control.

If the control already has a parent, SetParent removes the control from that parent's list of controls by calling the parent's RemoveControl method. If AParent is not nil (Delphi) or NULL (C++), SetParent calls its InsertControl method to add the control to its list of controls.

The TControl destructor calls SetParent(nil (Delphi) or NULL (C++)) to remove the control from its parent's control list before destroying the component.

See Also

Destroy

InsertControl

Parent

RemoveControl

63.1.1.4.12 TCustomRibbonComboBox.WMKillFocus Method

Pascal

```
procedure WMKillFocus(var Message: TWMKillFocus); message WM_KILLFOCUS;
```

C++

```
__fastcall WMKillFocus(TWMKillFocus Message);
```

Description

This is WMKillFocus, a member of class TCustomRibbonComboBox.

63.1.1.4.13 TCustomRibbonComboBox.WMSetFocus Method

Pascal

```
procedure WMSetFocus(var Message: TWMSetFocus); message WM_SETFOCUS;
```

C++

```
__fastcall WMSetFocus(TWMSetFocus Message);
```

Description

This is WMSetFocus, a member of class TCustomRibbonComboBox.

63.1.1.5 TCustomRibbonComboBox Properties

The properties of the TCustomRibbonComboBox class are listed here.

Public Properties

	Name	Description
	Color (see page 3591)	Specifies the background color of the control.
	Items (see page 3592)	This is Items, a member of class TCustomRibbonComboBox.
	MouseInControl (see page 3592)	This is MouseInControl, a member of class TCustomRibbonComboBox.

63.1.1.5.1 TCustomRibbonComboBox.Color Property

Specifies the background color of the control.

Pascal

```
property Color;
```

C++

```
__property Color;
```

Description

Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to false.

C++ Examples:

```
/*
The following code changes the color of a shape control
using the Color dialog box. The example displays the Color
dialog box when the Button1 button is clicked, allowing the
user to select a color with the dialog box. When a color
value is selected with the dialog box, it is assigned to
the shape control.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  if (ColorDialog1->Execute())
    Shape1->Brush->Color = ColorDialog1->Color;
}
```

Delphi Examples:

```
{
The following code changes the color of a shape control
using the Color dialog box. The example displays the Color
dialog box when the Button1 button is clicked, allowing the
user to select a color with the dialog box. When a color
value is selected with the dialog box, it is assigned to
the shape control.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  if ColorDialog1.Execute then
    Shape1.Brush.Color := ColorDialog1.Color;
end;
```

See Also

ParentColor

63.1.1.5.2 TCustomRibbonComboBox.Items Property

Pascal

```
property Items: TStrings;
```

C++

```
__property TStrings Items;
```

Description

This is Items, a member of class TCustomRibbonComboBox.

63.1.1.5.3 TCustomRibbonComboBox.MouseInControl Property

Pascal

```
property MouseInControl: Boolean;
```

C++

```
__property Boolean MouseInControl;
```

Description

This is MouseInControl, a member of class TCustomRibbonComboBox.

63.1.2 TKeyTipList Class

KeyTip list.

Class Hierarchy

RibbonActnCtrls.TKeyTipList

Pascal

```
TKeyTipList = class;
```

C++

```
class TKeyTipList;
```

File

RibbonActnCtrls

Description

TKeyTipList is a **KeyTip** list.

63.1.2.1 TKeyTipList Members

The following tables list the members exposed by TKeyTipList.

Public Methods

	Name	Description
	Add (see page 3594)	Add new match to list of KeyTips.
	Clear (see page 3594)	Clear list of KeyTips.

	Create (see page 3593)	Create TKeyTipList (see page 3592) instance.
V	Destroy (see page 3593)	Destroy TKeyTipList (see page 3592) instance.

Public Properties

	Name	Description
R	Count (see page 3594)	Number of valid **KeyTip**s.
	Current (see page 3595)	Current **KeyTip** value so far.
R	Full (see page 3595)	Indicates **KeyTip** has full match in list and no other matches
R	Items (see page 3595)	Indexable list of TKeyTipList (see page 3592).
R	Partial (see page 3596)	**KeyTip** has partial match in list.

63.1.2.2 TKeyTipList.Create Constructor

Create TKeyTipList (see page 3592) instance.

Pascal

```
constructor Create;
```

C++

```
__fastcall TKeyTipList();
```

Description

This method creates a TKeyTipList (see page 3592) instance.

See Also

Destroy (see page 3593)

63.1.2.3 TKeyTipList.Destroy Destructor

Destroy TKeyTipList (see page 3592) instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TKeyTipList();
```

Description

This method destroys a TKeyTipList (see page 3592) instance.

See Also

Create (see page 3593)

63.1.2.4 TKeyTipList Methods

The methods of the TKeyTipList class are listed here.

Public Methods

	Name	Description
	Add (see page 3594)	Add new match to list of KeyTips.
	Clear (see page 3594)	Clear list of KeyTips.

63.1.2.4.1 TKeyTipList.Add Method

Add new match to list of KeyTips.

Pascal

```
procedure Add(MatchType: TKeyTipMatchType; Handled: TKeyTipHandled; Control: TControl; Index:
Integer = -1);
```

C++

```
__fastcall Add(TKeyTipMatchType MatchType, TKeyTipHandled Handled, TControl * Control, int
Index = -1);
```

Description

This method adds a new match to the list of KeyTips.

See Also

Clear (see page 3594)

63.1.2.4.2 TKeyTipList.Clear Method

Clear list of KeyTips.

Pascal

```
procedure Clear;
```

C++

```
__fastcall Clear();
```

Description

This method clears the list of KeyTips.

See Also

Add (see page 3594)

63.1.2.5 TKeyTipList Properties

The properties of the TKeyTipList class are listed here.

Public Properties

	Name	Description
R	Count (see page 3594)	Number of valid **KeyTip**s.
	Current (see page 3595)	Current **KeyTip** value so far.
R	Full (see page 3595)	Indicates **KeyTip** has full match in list and no other matches
R	Items (see page 3595)	Indexable list of TKeyTipList (see page 3592).
R	Partial (see page 3596)	**KeyTip** has partial match in list.

63.1.2.5.1 TKeyTipList.Count Property

Number of valid **KeyTip**s.

Pascal

```
property Count: Integer;
```

C++

```
__property int Count;
```

Description

Count contains the number of valid **KeyTip**s.

See Also

Current (see page 3595)

63.1.2.5.2 TKeyTipList.Current Property

Current **KeyTip** value so far.

Pascal

```
property Current: string;
```

C++

```
__property UnicodeString Current;
```

Description

Current is the current **KeyTip** value the user has typed in so far.

See Also

Count (see page 3594)

63.1.2.5.3 TKeyTipList.Full Property

Indicates **KeyTip** has full match in list and no other matches

Pascal

```
property Full: Boolean;
```

C++

```
__property Boolean Full;
```

Description

Full is True if the current **KeyTip** has a full match in the list and there are no other matches, and it is False otherwise.

See Also

Partial (see page 3596)

63.1.2.5.4 TKeyTipList.Items Property

Indexable list of TKeyTipList (see page 3592).

Pascal

```
property Items [Index: Integer]: TKeyTipMatchItem;
```

C++

```
__property TKeyTipMatchItem Items[int Index];
```

Description

Items is an indexable list of TKeyTipList (see page 3592).

63.1.2.5.5 TKeyTipList.Partial Property

KeyTip has partial match in list.

Pascal

```
property Partial: Boolean;
```

C++

```
__property Boolean Partial;
```

Description

Partial is True if the current **KeyTip** has a partial match in the list and false otherwise.

See Also

Full (see page 3595)

63.1.3 TKeyTipListEnumerator Class

Class Hierarchy

Pascal

```
TKeyTipListEnumerator = class;
```

C++

```
class TKeyTipListEnumerator;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TKeyTipListEnumerator.

63.1.3.1 TKeyTipListEnumerator Members

The following tables list the members exposed by TKeyTipListEnumerator.

Public Methods

	Name	Description
	Create (see page 3597)	This is Create, a member of class TKeyTipListEnumerator.
	GetCurrent (see page 3597)	This is GetCurrent, a member of class TKeyTipListEnumerator.
	MoveNext (see page 3597)	This is MoveNext, a member of class TKeyTipListEnumerator.

Public Properties

	Name	Description
R	Current (see page 3598)	This is Current, a member of class TKeyTipListEnumerator.

63.1.3.2 TKeyTipListEnumerator.Create Constructor

Pascal

```
constructor Create(AList: TKeyTipList);
```

C++

```
__fastcall TKeyTipListEnumerator(TKeyTipList AList);
```

Description

This is Create, a member of class TKeyTipListEnumerator.

63.1.3.3 TKeyTipListEnumerator Methods

The methods of the TKeyTipListEnumerator class are listed here.

Public Methods

	Name	Description
	GetCurrent (see page 3597)	This is GetCurrent, a member of class TKeyTipListEnumerator.
	MoveNext (see page 3597)	This is MoveNext, a member of class TKeyTipListEnumerator.

63.1.3.3.1 TKeyTipListEnumerator.GetCurrent Method

Pascal

```
function GetCurrent: TKeyTipMatchItem;
```

C++

```
__fastcall TKeyTipMatchItem GetCurrent();
```

Description

This is GetCurrent, a member of class TKeyTipListEnumerator.

63.1.3.3.2 TKeyTipListEnumerator.MoveNext Method

Pascal

```
function MoveNext: Boolean;
```

C++

```
__fastcall Boolean MoveNext();
```

Description

This is MoveNext, a member of class TKeyTipListEnumerator.

63.1.3.4 TKeyTipListEnumerator Properties

The properties of the TKeyTipListEnumerator class are listed here.

Public Properties

	Name	Description
	Current (see page 3598)	This is Current, a member of class TKeyTipListEnumerator.

63.1.3.4.1 TKeyTipListEnumerator.Current Property

Pascal

```
property Current: TKeyTipMatchItem;
```

C++

```
__property TKeyTipMatchItem Current;
```

Description

This is Current, a member of class TKeyTipListEnumerator.

63.1.4 TKeyTipMatchItem Class

Details about Full and Partial **KeyTip** matches.

Class Hierarchy

Pascal

```
TKeyTipMatchItem = class;
```

C++

```
class TKeyTipMatchItem;
```

File

RibbonActnCtrls

Description

TKeyTipMatchItem stores details about Full and Partial **KeyTip** matches. It contains a reference to the control that needs to be executed. If the MatchType (see page 3600) is **mtTabs**, then the Index (see page 3600) property is used to determine which Tab to change to.

See Also

Index (see page 3600)

MatchType (see page 3600)

TKeyTipMatchType (see page 3927)

63.1.4.1 TKeyTipMatchItem Members

The following tables list the members exposed by TKeyTipMatchItem.

Public Methods

	Name	Description
	Create (see page 3599)	This is Create, a member of class TKeyTipMatchItem.

Public Properties

	Name	Description
	Control (see page 3599)	This is Control, a member of class TKeyTipMatchItem.

	Handled (see page 3599)	This is Handled, a member of class TKeyTipMatchItem.
	Index (see page 3600)	Determines which Tab to change to.
	MatchType (see page 3600)	**KeyTip** match type.

63.1.4.2 TKeyTipMatchItem.Create Constructor

Pascal

```
constructor Create(AType: TKeyTipMatchType; AHandled: TKeyTipHandled; AControl: TControl;
AIndex: Integer = -1);
```

C++

```
__fastcall TKeyTipMatchItem(TKeyTipMatchType AType, TKeyTipHandled AHandled, TControl *
AControl, int AIndex = -1);
```

Description

This is Create, a member of class TKeyTipMatchItem.

63.1.4.3 TKeyTipMatchItem Properties

The properties of the TKeyTipMatchItem class are listed here.

Public Properties

	Name	Description
	Control (see page 3599)	This is Control, a member of class TKeyTipMatchItem.
	Handled (see page 3599)	This is Handled, a member of class TKeyTipMatchItem.
	Index (see page 3600)	Determines which Tab to change to.
	MatchType (see page 3600)	**KeyTip** match type.

63.1.4.3.1 TKeyTipMatchItem.Control Property

Pascal

```
property Control: TControl;
```

C++

```
__property TControl * Control;
```

Description

This is Control, a member of class TKeyTipMatchItem.

63.1.4.3.2 TKeyTipMatchItem.Handled Property

Pascal

```
property Handled: TKeyTipHandled;
```

C++

```
__property TKeyTipHandled Handled;
```

Description

This is Handled, a member of class TKeyTipMatchItem.

63.1.4.3.3 TKeyTipMatchItem.Index Property

Determines which Tab to change to.

Pascal

```
property Index: Integer;
```

C++

```
__property int Index;
```

Description

If the MatchType (see page 3600) is **mtTabs**, then the Index property is used to determine which Tab to change to.

See Also

MatchType (see page 3600)

TKeyTipMatchType (see page 3927)

63.1.4.3.4 TKeyTipMatchItem.MatchType Property

KeyTip match type.

Pascal

```
property MatchType: TKeyTipMatchType;
```

C++

```
__property TKeyTipMatchType MatchType;
```

Description

MatchType is the **KeyTip** match type.

See Also

Index (see page 3600)

TKeyTipMatchType (see page 3927)

63.1.5 TKeyTipValidator Class

Used to validate **KeyTip**.

Class Hierarchy

Pascal

```
TKeyTipValidator = class;
```

C++

```
class TKeyTipValidator;
```

File

RibbonActnCtrls

Description

TKeyTipValidator is a static class used to validate a **KeyTip**.

63.1.5.1 TKeyTipValidator Members

The following tables list the members exposed by TKeyTipValidator.

Public Methods

	Name	Description
	Validate (see page 3601)	This is Validate, a member of class TKeyTipValidator.

63.1.5.2 TKeyTipValidator Methods

The methods of the TKeyTipValidator class are listed here.

Public Methods

	Name	Description
	Validate (see page 3601)	This is Validate, a member of class TKeyTipValidator.

63.1.5.2.1 TKeyTipValidator.Validate Method

Pascal

```
class procedure Validate(const SrcKeyTip: string; const CmdKeyTip: string; Control: TControl;
var Handled: TKeyTipHandled);
```

C++

```
__fastcall Validate(const UnicodeString SrcKeyTip, const UnicodeString CmdKeyTip, TControl *
Control, TKeyTipHandled Handled);
```

Description

This is Validate, a member of class TKeyTipValidator.

63.1.6 TRibbonAddRemoveItem Class

Class Hierarchy

Pascal

```
TRibbonAddRemoveItem = class(TXPStyleAddRemoveItem);
```

C++

```
class TRibbonAddRemoveItem : public TXPStyleAddRemoveItem;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonAddRemoveItem.

63.1.6.1 TRibbonAddRemoveItem Members

The following tables list the members exposed by TRibbonAddRemoveItem.

Protected Methods

	Name	Description
	DrawBackground (see page 3602)	Draws a background image on the control.
	DrawGlyph (see page 3602)	Draws a symbol on the control.

63.1.6.2 TRibbonAddRemoveItem Methods

The methods of the TRibbonAddRemoveItem class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3602)	Draws a background image on the control.
	DrawGlyph (see page 3602)	Draws a symbol on the control.

63.1.6.2.1 TRibbonAddRemoveItem.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.6.2.2 TRibbonAddRemoveItem.DrawGlyph Method

Draws a symbol on the control.

Pascal

```
procedure DrawGlyph(const Location: TPoint); override;
```

C++

```
virtual __fastcall DrawGlyph(const TPoint Location);
```

Description

Use DrawGlyph to draw a symbol on the control at the specified location.

See Also

DrawBackground

DrawText

63.1.7 TRibbonApplicationButton Class

Class Hierarchy

Pascal

```
TRibbonApplicationButton = class(TCustomMenuButton);
```

C++

```
class TRibbonApplicationButton : public TCustomMenuButton;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonApplicationButton.

63.1.7.1 TRibbonApplicationButton Members

The following tables list the members exposed by TRibbonApplicationButton.

Public Methods

	Name	Description
	CalcBounds (see page 3603)	Calculates the size of the control.

63.1.7.2 TRibbonApplicationButton Methods

The methods of the TRibbonApplicationButton class are listed here.

Public Methods

	Name	Description
	CalcBounds (see page 3603)	Calculates the size of the control.

63.1.7.2.1 TRibbonApplicationButton.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.8 TRibbonApplicationMenuButton Class

Application Button when using Ribbon (see page 3347) interface.

Class Hierarchy

Pascal

```
TRibbonApplicationMenuButton = class(TCustomMenuButton, IRibbonKeyTip);
```

C++

```
class TRibbonApplicationMenuButton : public TCustomMenuButton, public IRibbonKeyTip;
```

File

RibbonActnCtrls

Description

TRibbonApplicationMenuButton is used for the Application Button when using the Ribbon (see page 3347) interface.

63.1.8.1 TRibbonApplicationMenuButton Members

The following tables list the members exposed by TRibbonApplicationMenuButton.

Protected Methods

	Name	Description
	CMMouseEnter (see page 3607)	This is CMMouseEnter, a member of class TRibbonApplicationMenuButton.
	CMMouseLeave (see page 3607)	This is CMMouseLeave, a member of class TRibbonApplicationMenuButton.
	DragOver (see page 3607)	OnDragOver event dispatcher.
	Paint (see page 3609)	Render control surface.
	RibbonStyle (see page 3609)	This is RibbonStyle, a member of class TRibbonApplicationMenuButton.
	SetParent (see page 3610)	Sets the parent of the control.
	UMFadeControl (see page 3611)	This is UMFadeControl, a member of class TRibbonApplicationMenuButton.
	WMLButtonDblClk (see page 3611)	This is WMLButtonDblClk, a member of class TRibbonApplicationMenuButton.

Protected Properties

	Name	Description
	MouseInControl (see page 3612)	This is MouseInControl, a member of class TRibbonApplicationMenuButton.

Public Methods

	Name	Description
	CalcBounds (see page 3606)	Calculates the size of the control.
	Create (see page 3605)	Creates an instance of TCustomMenuButton.
	Destroy (see page 3605)	Destroys an instance of TCustomMenuButton.
	GetKeyTip (see page 3607)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3608)	Indicates if command's **KeyTip** matches currently input KeyTip.

	HideKeyTip (see page 3608)	Hide **KeyTip** for command.
	KeyTipClick (see page 3609)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3610)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3611)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3612)	Current **KeyTip** for command

63.1.8.2 TRibbonApplicationMenuButton.Create Constructor

Creates an instance of TCustomMenuButton.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonApplicationMenuButton(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomMenuButton. However, you should never attempt to instantiate a TCustomMenuButton. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

AOwner is the component that is responsible for freeing the menu button instance. It becomes the value of the Owner property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Owner

Create

Destroy

63.1.8.3 TRibbonApplicationMenuButton.Destroy Destructor

Destroys an instance of TCustomMenuButton.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRibbonApplicationMenuButton();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the graphic control reference is not nil, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

As the control is destroyed, it:

Frees the TCanvas object in its Canvas property.

Override Destroy to free any memory or resources allocated in the Create method. When declaring a Destroy method in a descendant control type, always add the override directive to the declaration and call the inherited Destroy as the last statement in the redeclared method.

When a control is created, Delphi automatically provides exception handling around the constructor call. The destructor is called if an exception escapes from the constructor. This means that the destructor code must be prepared to clean up a partially constructed instance. Check all data fields for zero before disposing of their contents.

See Also

Create

System::TObject::Free

63.1.8.4 TRibbonApplicationMenuButton Methods

The methods of the TRibbonApplicationMenuButton class are listed here.

Protected Methods

	Name	Description
	CMMouseEnter (see page 3607)	This is CMMouseEnter, a member of class TRibbonApplicationMenuButton.
	CMMouseLeave (see page 3607)	This is CMMouseLeave, a member of class TRibbonApplicationMenuButton.
	DragOver (see page 3607)	OnDragOver event dispatcher.
	Paint (see page 3609)	Render control surface.
	RibbonStyle (see page 3609)	This is RibbonStyle, a member of class TRibbonApplicationMenuButton.
	SetParent (see page 3610)	Sets the parent of the control.
	UMFadeControl (see page 3611)	This is UMFadeControl, a member of class TRibbonApplicationMenuButton.
	WMLButtonDblClk (see page 3611)	This is WMLButtonDblClk, a member of class TRibbonApplicationMenuButton.

Public Methods

	Name	Description
	CalcBounds (see page 3606)	Calculates the size of the control.
	GetKeyTip (see page 3607)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3608)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3608)	Hide **KeyTip** for command.
	KeyTipClick (see page 3609)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3610)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3611)	Shows **KeyTip** for command in specified position.

63.1.8.4.1 TRibbonApplicationMenuButton.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.8.4.2 TRibbonApplicationMenuButton.CMMouseEnter Method

Pascal

```
procedure CMMouseEnter(var Message: TMessage); override;
```

C++

```
virtual __fastcall CMMouseEnter(TMessage Message);
```

Description

This is CMMouseEnter, a member of class TRibbonApplicationMenuButton.

63.1.8.4.3 TRibbonApplicationMenuButton.CMMouseLeave Method

Pascal

```
procedure CMMouseLeave(var Message: TMessage); override;
```

C++

```
virtual __fastcall CMMouseLeave(TMessage Message);
```

Description

This is CMMouseLeave, a member of class TRibbonApplicationMenuButton.

63.1.8.4.4 TRibbonApplicationMenuButton.DragOver Method

OnDragOver event dispatcher.

Pascal

```
procedure DragOver(Source: TObject; X: Integer; Y: Integer; State: TDragState; var Accept:
Boolean); override;
```

C++

```
virtual __fastcall DragOver(TObject * Source, int X, int Y, TDragState State, Boolean Accept);
```

Description

Override DragOver to add additional code that executes before the OnDragOver event handler is called.

DragOver sets the Accept parameter to true to indicate that the user can drop the dragged object on the control. It sets Accept to false to indicate that the user cannot drop the dragged object on the control.

The Source parameter is the object being dragged. The State parameter indicates how the dragged object is moving in relation to the control. X and Y indicate the current position of the mouse.

See Also

DragDrop

OnDragOver

63.1.8.4.5 TRibbonApplicationMenuButton.GetKeyTip Method

Returns current **KeyTip** for command.

Pascal

```
function GetKeyTip: string;
```

C++

```
__fastcall UnicodeString GetKeyTip();
```

Description

GetKeyTip returns the current **KeyTip** for the command

See Also

KeyTip (see page 3922)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.8.4.6 TRibbonApplicationMenuButton.HandleKeyTip Method

Indicates if command's **KeyTip** matches currently input KeyTip.

Pascal

```
procedure HandleKeyTip(const KeyTip: string; var Handled: TKeyTipHandled);
```

C++

```
__fastcall HandleKeyTip(const UnicodeString KeyTip, TKeyTipHandled Handled);
```

Description

HandleKeyTip indicates whether this command's

KeyTip matches the currently input KeyTip by the user. Full and partial matches are taken into consideration.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.8.4.7 TRibbonApplicationMenuButton.HideKeyTip Method

Hide **KeyTip** for command.

Pascal

```
procedure HideKeyTip(const Match: string);
```

C++

```
__fastcall HideKeyTip(const UnicodeString Match);
```

Description

This method hides the **KeyTip** for this command. If the **Match** parameter matches (full or partial) the command's **KeyTip**, the **KeyTip** remains visible.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.8.4.8 TRibbonApplicationMenuButton.KeyTipClick Method

Handle clicking a **KeyTip**.

Pascal

```
procedure KeyTipClick;
```

C++

```
__fastcall KeyTipClick();
```

Description

KeyTipClick Handle clicking a **KeyTip**. Some requires need to be handled differently when clicked by a **KeyTip**.

63.1.8.4.9 TRibbonApplicationMenuButton.Paint Method

Render control surface.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method is called automatically when a lightweight control needs update its display area. Code in the Paint method uses the Canvas property to render the control surface.

As implemented in TGraphicControl, Paint does nothing. Specific rendering logic is provided by Paint methods in descendant control classes.

See Also

Canvas

63.1.8.4.10 TRibbonApplicationMenuButton.RibbonStyle Method

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

This is RibbonStyle, a member of class TRibbonApplicationMenuButton.

63.1.8.4.11 TRibbonApplicationMenuButton.SetKeyTip Method

Sets **KeyTip** for command.

Pascal

```
procedure SetKeyTip(const Value: string);
```

C++

```
__fastcall SetKeyTip(const UnicodeString Value);
```

Description

This method sets the **KeyTip** for the command.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

ShowKeyTip (see page 3921)

63.1.8.4.12 TRibbonApplicationMenuButton.SetParent Method

Sets the parent of the control.

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

SetParent is the protected implementation of the Parent property. Override SetParent to execute additional code when the value of the Parent property changes.

The AParent parameter specifies the new parent of the control.

If the control already has a parent, SetParent removes the control from that parent's list of controls by calling the parent's RemoveControl method. If AParent is not nil (Delphi) or NULL (C++), SetParent calls its InsertControl method to add the control to its list of controls.

The TControl destructor calls SetParent(nil (Delphi) or NULL (C++)) to remove the control from its parent's control list before destroying the component.

See Also

Destroy

InsertControl

Parent

RemoveControl

63.1.8.4.13 TRibbonApplicationMenuButton.ShowKeyTip Method

Shows **KeyTip** for command in specified position.

Pascal

```
procedure ShowKeyTip(const Position: TKeyTipPosition);
```

C++

```
__fastcall ShowKeyTip(const TKeyTipPosition Position);
```

Description

This method shows the **KeyTip** for the command in the position specified. Only the Application Button uses a **KeyTip** position.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

63.1.8.4.14 TRibbonApplicationMenuButton.UMFadeControl Method

Pascal

```
procedure UMFadeControl(var Message: TMessage); message UM_FADECONTROL;
```

C++

```
__fastcall UMFadeControl(TMessage Message);
```

Description

This is UMFadeControl, a member of class TRibbonApplicationMenuButton.

63.1.8.4.15 TRibbonApplicationMenuButton.WMLButtonDblClk Method

Pascal

```
procedure WMLButtonDblClk(var Message: TWMLButtonDblClk); message WM_LBUTTONDBLCLK;
```

C++

```
__fastcall WMLButtonDblClk(TWMLButtonDblClk Message);
```

Description

This is WMLButtonDblClk, a member of class TRibbonApplicationMenuButton.

63.1.8.5 TRibbonApplicationMenuButton Properties

The properties of the TRibbonApplicationMenuButton class are listed here.

Protected Properties

	Name	Description
	MouseInControl (see page 3612)	This is MouseInControl, a member of class TRibbonApplicationMenuButton.

Public Properties

	Name	Description
	KeyTip (see page 3612)	Current **KeyTip** for command

63.1.8.5.1 TRibbonApplicationMenuButton.KeyTip Property

Current **KeyTip** for command

Pascal

```
property KeyTip: string;
```

C++

```
__property UnicodeString KeyTip;
```

Description

KeyTip is the current **KeyTip** for the command

See Also

GetKeyTip (see page 3919)

SetKeyTip (see page 3921)

63.1.8.5.2 TRibbonApplicationMenuButton.MouseInControl Property

Pascal

```
property MouseInControl: Boolean;
```

C++

```
__property Boolean MouseInControl;
```

Description

This is MouseInControl, a member of class TRibbonApplicationMenuButton.

63.1.9 TRibbonApplicationMenuItem Class

Class Hierarchy

Pascal

```
TRibbonApplicationMenuItem = class(TRibbonMenuItem);
```

C++

```
class TRibbonApplicationMenuItem : public TRibbonMenuItem;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonApplicationMenuItem.

63.1.9.1 TRibbonApplicationMenuItem Members

The following tables list the members exposed by TRibbonApplicationMenuItem.

Protected Methods

	Name	Description
	CalcLayout (see page 3757)	This is CalcLayout, a member of class TRibbonMenuItem.
	DrawBackground (see page 3758)	Draws a background image on the control.
	DrawGlyph (see page 3614)	Draws a symbol on the control.
	DrawSelectedBackground (see page 3615)	This is DrawSelectedBackground, a member of class TRibbonApplicationMenuItem.
	DrawSeparator (see page 3615)	This is DrawSeparator, a member of class TRibbonApplicationMenuItem.
	DrawShadowedText (see page 3615)	This is DrawShadowedText, a member of class TRibbonApplicationMenuItem.
	DrawSubMenuGlyph (see page 3615)	This is DrawSubMenuGlyph, a member of class TRibbonApplicationMenuItem.
	DrawText (see page 3616)	Draws text on the control.
	GetImageSize (see page 3616)	This is GetImageSize, a member of class TRibbonApplicationMenuItem.
	GetShowShortCut (see page 3759)	This is GetShowShortCut, a member of class TRibbonMenuItem.
	RibbonStyle (see page 3760)	This is RibbonStyle, a member of class TRibbonMenuItem.

Protected Properties

	Name	Description
	DrawBanner (see page 3760)	This is DrawBanner, a member of class TRibbonMenuItem.

Public Methods

	Name	Description
	CalcBounds (see page 3614)	Calculates the size of the menu item.
	Click (see page 3757)	Executes the control's function.
	Create (see page 3613)	Creates an instance of TCustomMenuItem.

63.1.9.2 TRibbonApplicationMenuItem.Create Constructor

Creates an instance of TCustomMenuItem.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonApplicationMenuItem(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomMenuItem. However, you should never attempt to instantiate a TCustomMenuItem. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

After calling the inherited constructor, Create initializes the control, including the following property settings:

Initializes the TCanvas object for its Canvas property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Create

Destroy

63.1.9.3 TRibbonApplicationMenuItem Methods

The methods of the TRibbonApplicationMenuItem class are listed here.

Protected Methods

	Name	Description
	DrawGlyph (see page 3614)	Draws a symbol on the control.
	DrawSelectedBackground (see page 3615)	This is DrawSelectedBackground, a member of class TRibbonApplicationMenuItem.
	DrawSeparator (see page 3615)	This is DrawSeparator, a member of class TRibbonApplicationMenuItem.
	DrawShadowedText (see page 3615)	This is DrawShadowedText, a member of class TRibbonApplicationMenuItem.
	DrawSubMenuGlyph (see page 3615)	This is DrawSubMenuGlyph, a member of class TRibbonApplicationMenuItem.
	DrawText (see page 3616)	Draws text on the control.
	GetImageSize (see page 3616)	This is GetImageSize, a member of class TRibbonApplicationMenuItem.

Public Methods

	Name	Description
	CalcBounds (see page 3614)	Calculates the size of the menu item.

63.1.9.3.1 TRibbonApplicationMenuItem.CalcBounds Method

Calculates the size of the menu item.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as whether or not the menu item is a separator or not. CalcBounds computes the boundaries of the menu item considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.9.3.2 TRibbonApplicationMenuItem.DrawGlyph Method

Draws a symbol on the control.

Pascal

```
procedure DrawGlyph(const Location: TPoint); override;
```

C++

```
virtual __fastcall DrawGlyph(const TPoint Location);
```

Description

Use DrawGlyph to draw a symbol on the control at the specified location.

See Also

DrawBackground

DrawText

63.1.9.3.3 TRibbonApplicationMenuItem.DrawSelectedBackground Method

Pascal

```
procedure DrawSelectedBackground(ARect: TRect); override;
```

C++

```
virtual __fastcall DrawSelectedBackground(TRect ARect);
```

Description

This is DrawSelectedBackground, a member of class TRibbonApplicationMenuItem.

63.1.9.3.4 TRibbonApplicationMenuItem.DrawSeparator Method

Pascal

```
procedure DrawSeparator(const Offset: Integer); override;
```

C++

```
virtual __fastcall DrawSeparator(const int Offset);
```

Description

This is DrawSeparator, a member of class TRibbonApplicationMenuItem.

63.1.9.3.5 TRibbonApplicationMenuItem.DrawShadowedText Method

Pascal

```
procedure DrawShadowedText(Rect: TRect; Flags: Cardinal; Text: string; TextColor: TColor;
ShadowColor: TColor); override;
```

C++

```
virtual __fastcall DrawShadowedText(TRect Rect, unsigned Flags, UnicodeString Text, TColor
TextColor, TColor ShadowColor);
```

Description

This is DrawShadowedText, a member of class TRibbonApplicationMenuItem.

63.1.9.3.6 TRibbonApplicationMenuItem.DrawSubMenuGlyph Method

Pascal

```
procedure DrawSubMenuGlyph; override;
```

C++

```
virtual __fastcall DrawSubMenuGlyph();
```

Description

This is DrawSubMenuGlyph, a member of class TRibbonApplicationMenuItem.

63.1.9.3.7 TRibbonApplicationMenuItem.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var ARect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect ARect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

See Also

DrawBackground

DrawGlyph

63.1.9.3.8 TRibbonApplicationMenuItem.GetImageSize Method

Pascal

```
function GetImageSize: TPoint; override;
```

C++

```
virtual __fastcall TPoint GetImageSize();
```

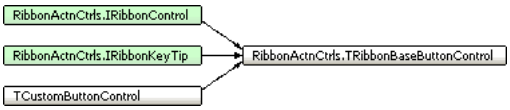
Description

This is GetImageSize, a member of class TRibbonApplicationMenuItem.

63.1.10 TRibbonBaseButtonControl Class

Common ancestor for many buttons in groups and Quick Access Toolbar.

Class Hierarchy



Pascal

```
TRibbonBaseButtonControl = class(TCustomButtonControl, IRibbonControl, IRibbonKeyTip);
```

C++

```
class TRibbonBaseButtonControl : public TCustomButtonControl, public IRibbonControl, public
IRibbonKeyTip;
```

File

RibbonActnCtrls

Description

TRibbonBaseButtonControl is the common ancestor for many buttons that appear in the groups and on the Quick Access Toolbar.

63.1.10.1 TRibbonBaseButtonControl Members

The following tables list the members exposed by TRibbonBaseButtonControl.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	DrawBackground (see page 3621)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawText (see page 3622)	Draws text on the control.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3619)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.
	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.10.2 TRibbonBaseButtonControl.Create Constructor

Creates an instance of TCustomButtonControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonBaseButtonControl(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomButtonControl. However, you should never attempt to instantiate a TCustomButtonControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

AOwner is the component that is responsible for freeing the button control. It becomes the value of the Owner property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Owner

Create

Destroy

63.1.10.3 TRibbonBaseButtonControl.Destroy Destructor

Destroys an instance of TCustomActionControl.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRibbonBaseButtonControl();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the graphic control reference is not nil, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

As the control is destroyed, it:

Frees the TCanvas object in its Canvas property.

Override Destroy to free any memory or resources allocated in the Create method. When declaring a Destroy method in a descendant control type, always add the override directive to the declaration and call the inherited Destroy as the last statement in the redeclared method.

When a control is created, Delphi automatically provides exception handling around the constructor call. The destructor is called if an exception escapes from the constructor. This means that the destructor code must be prepared to clean up a partially constructed instance. Check all data fields for zero before disposing of their contents.

See Also

Create

System::TObject::Free

63.1.10.4 TRibbonBaseButtonControl Methods

The methods of the TRibbonBaseButtonControl class are listed here.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	DrawBackground (see page 3621)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawText (see page 3622)	Draws text on the control.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3619)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

63.1.10.4.1 TRibbonBaseButtonControl.AllowHotKeys Method

Pascal

```
function AllowHotKeys: Boolean; virtual;
```

C++

```
virtual __fastcall Boolean AllowHotKeys();
```

Description

This is AllowHotKeys, a member of class TRibbonBaseButtonControl.

63.1.10.4.2 TRibbonBaseButtonControl.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.10.4.3 TRibbonBaseButtonControl.CanAddToQuickAccessToolbar Method

Pascal

```
function CanAddToQuickAccessToolbar: Boolean;
```

C++

```
__fastcall Boolean CanAddToQuickAccessToolbar();
```

Description

This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.

63.1.10.4.4 TRibbonBaseButtonControl.Click Method

Executes the control's function.

Pascal

```
procedure Click; override;
```

C++

```
virtual __fastcall Click();
```

Description

This override of the method inherited from TCustomActionControl is almost identical to the inherited method. The only difference is that if the button control is being manipulated at design time, extra actions are taken to ensure that the control is properly selected.

See Also

Click

Click

UsageCount

63.1.10.4.5 TRibbonBaseButtonControl.DragDrop Method

Performs actions and generates an OnDragDrop event.

Pascal

```
procedure DragDrop(Source: TObject; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall DragDrop(TObject * Source, int X, int Y);
```

Description

This override of the method inherited from TControl executes instructions and generates an OnDragDrop event. The method is

called when a user drags and drops an item (Source) onto the custom action control at coordinates (X,Y).

See Also

DragDrop

63.1.10.4.6 TRibbonBaseButtonControl.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.10.4.7 TRibbonBaseButtonControl.DrawGlyph Method

Draws a symbol on the control.

Pascal

```
procedure DrawGlyph(const Location: TPoint); override;
```

C++

```
virtual __fastcall DrawGlyph(const TPoint Location);
```

Description

Use DrawGlyph to draw a symbol on the control at the specified location.

See Also

DrawBackground

DrawText

63.1.10.4.8 TRibbonBaseButtonControl.DrawSeparator Method

Pascal

```
procedure DrawSeparator(const Offset: Integer); override;
```

C++

```
virtual __fastcall DrawSeparator(const int Offset);
```

Description

This is DrawSeparator, a member of class TRibbonBaseButtonControl.

63.1.10.4.9 TRibbonBaseButtonControl.DrawShadowedText Method

Pascal

```
procedure DrawShadowedText(Rect: TRect; Flags: Cardinal; Text: string; TextColor: TColor;
ShadowColor: TColor); override;
```

C++

```
virtual __fastcall DrawShadowedText(TRect Rect, unsigned Flags, UnicodeString Text, TColor
TextColor, TColor ShadowColor);
```

Description

This is DrawShadowedText, a member of class TRibbonBaseButtonControl.

63.1.10.4.10 TRibbonBaseButtonControl.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var ARect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect ARect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

See Also

DrawBackground

DrawGlyph

63.1.10.4.11 TRibbonBaseButtonControl.GetImageSize Method

Pascal

```
function GetImageSize: TPoint; override;
```

C++

```
virtual __fastcall TPoint GetImageSize();
```

Description

This is GetImageSize, a member of class TRibbonBaseButtonControl.

63.1.10.4.12 TRibbonBaseButtonControl.GetKeyTip Method

Returns current **KeyTip** for command.

Pascal

```
function GetKeyTip: string;
```

C++

```
__fastcall UnicodeString GetKeyTip();
```

Description

GetKeyTip returns the current **KeyTip** for the command

See Also

KeyTip (see page 3922)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.10.4.13 TRibbonBaseButtonControl.HandleKeyTip Method

Indicates if command's **KeyTip** matches currently input KeyTip.

Pascal

```
procedure HandleKeyTip(const KeyTip: string; var Handled: TKeyTipHandled);
```

C++

```
__fastcall HandleKeyTip(const UnicodeString KeyTip, TKeyTipHandled Handled);
```

Description

HandleKeyTip indicates whether this command's

KeyTip matches the currently input KeyTip by the user. Full and partial matches are taken into consideration.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.10.4.14 TRibbonBaseButtonControl.HideKeyTip Method

Hide **KeyTip** for command.

Pascal

```
procedure HideKeyTip(const Match: string);
```

C++

```
__fastcall HideKeyTip(const UnicodeString Match);
```

Description

This method hides the **KeyTip** for this command. If the **Match** parameter matches (full or partial) the command's **KeyTip**, the **KeyTip** remains visible.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.10.4.15 TRibbonBaseButtonControl.IsButton Method

Pascal

```
function IsButton(out ButtonProperties: TButtonProperties): Boolean;
```

C++

```
__fastcall Boolean IsButton(TButtonProperties ButtonProperties);
```

Description

This is IsButton, a member of class TRibbonBaseButtonControl.

63.1.10.4.16 TRibbonBaseButtonControl.IsRibbonAvailable Method

Pascal

```
function IsRibbonAvailable: Boolean;
```

C++

```
__fastcall Boolean IsRibbonAvailable();
```

Description

This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.

63.1.10.4.17 TRibbonBaseButtonControl.KeyTipClick Method

Handle clicking a **KeyTip**.

Pascal

```
procedure KeyTipClick; virtual;
```

C++

```
virtual __fastcall KeyTipClick();
```

Description

KeyTipClick Handle clicking a **KeyTip**. Some requires need to be handled differently when clicked by a **KeyTip**.

63.1.10.4.18 TRibbonBaseButtonControl.RibbonStyle Method

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

This is RibbonStyle, a member of class TRibbonBaseButtonControl.

63.1.10.4.19 TRibbonBaseButtonControl.SetKeyTip Method

Sets **KeyTip** for command.

Pascal

```
procedure SetKeyTip(const Value: string);
```

C++

```
__fastcall SetKeyTip(const UnicodeString Value);
```

Description

This method sets the **KeyTip** for the command.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

ShowKeyTip (see page 3921)

63.1.10.4.20 TRibbonBaseButtonControl.ShowKeyTip Method

Shows **KeyTip** for command in specified position.

Pascal

```
procedure ShowKeyTip(const Position: TKeyTipPosition);
```

C++

```
__fastcall ShowKeyTip(const TKeyTipPosition Position);
```

Description

This method shows the **KeyTip** for the command in the position specified. Only the Application Button uses a **KeyTip** position.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

63.1.10.5 TRibbonBaseButtonControl Properties

The properties of the TRibbonBaseButtonControl class are listed here.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

63.1.10.5.1 TRibbonBaseButtonControl.DrawDown Property

Pascal

```
property DrawDown: Boolean;
```

C++

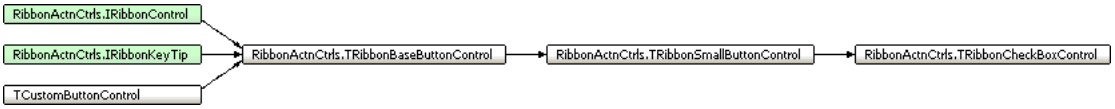
```
__property Boolean DrawDown;
```

Description

This is DrawDown, a member of class TRibbonBaseButtonControl.

63.1.11 TRibbonCheckBoxControl Class

Class Hierarchy



Pascal

```
TRibbonCheckBoxControl = class(TRibbonSmallButtonControl);
```

C++

```
class TRibbonCheckBoxControl : public TRibbonSmallButtonControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonCheckBoxControl.

63.1.11.1 TRibbonCheckBoxControl Members

The following tables list the members exposed by TRibbonCheckBoxControl.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	DrawBackground (see page 3628)	Draws a background image on the control.
	DrawGlyph (see page 3628)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawText (see page 3622)	Draws text on the control.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3627)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.
	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.11.2 TRibbonCheckBoxControl Methods

The methods of the TRibbonCheckBoxControl class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3628)	Draws a background image on the control.
	DrawGlyph (see page 3628)	Draws a symbol on the control.

Public Methods

	Name	Description
	CalcBounds (see page 3627)	Calculates the size of the control.

63.1.11.2.1 TRibbonCheckBoxControl.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.11.2.2 TRibbonCheckBoxControl.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.11.2.3 TRibbonCheckBoxControl.DrawGlyph Method

Draws a symbol on the control.

Pascal

```
procedure DrawGlyph(const Location: TPoint); override;
```

C++

```
virtual __fastcall DrawGlyph(const TPoint Location);
```

Description

Use DrawGlyph to draw a symbol on the control at the specified location.

See Also

DrawBackground

DrawText

63.1.12 TRibbonComboBox Class

Class Hierarchy

Pascal

```
TRibbonComboBox = class(TCustomRibbonComboBox);
```

C++

```
class TRibbonComboBox : public TCustomRibbonComboBox;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonComboBox.

63.1.12.1 TRibbonComboBox Members

The following tables list the members exposed by TRibbonComboBox.

Protected Methods

	Name	Description
	BeginAutoDrag (see page 3585)	Begin Automatic Drag.
	CMMouseenter (see page 3585)	This is CMMouseenter, a member of class TCustomRibbonComboBox.
	CMMouseleave (see page 3586)	This is CMMouseleave, a member of class TCustomRibbonComboBox.
	DragOver (see page 3586)	OnDragOver event dispatcher.
	KeyDown (see page 3587)	Respond to key press events.
	KeyPress (see page 3587)	Respond to keyboard input.
	MouseMove (see page 3588)	Respond to mouse moving over control area..
	Notification (see page 3588)	Responds to notifications that components are being created or destroyed.
	SetParent (see page 3589)	Sets the parent of the control.
	WMKillFocus (see page 3590)	This is WMKillFocus, a member of class TCustomRibbonComboBox.
	WMSetFocus (see page 3590)	This is WMSetFocus, a member of class TCustomRibbonComboBox.

Public Methods

	Name	Description
	Create (see page 3584)	Creates an instance of TCustomEdit.
	Destroy (see page 3584)	Destroys an instance of TWinControl.
	DragDrop (see page 3586)	OnDragDrop event dispatcher.
	SetBounds (see page 3589)	Sets the windowed control's boundary properties all at once.

Public Properties

	Name	Description
	Color (see page 3591)	Specifies the background color of the control.
	MouseInControl (see page 3592)	This is MouseInControl, a member of class TCustomRibbonComboBox.

Published Events

	Name	Description
	OnChange (see page 3648)	Occurs when the text for the edit control may have changed.
	OnClick (see page 3649)	Occurs when the user clicks the control.
	OnContextPopup (see page 3651)	Occurs when the user right-clicks the control or otherwise invokes the popup menu (such as using the keyboard).
	OnDblClick (see page 3651)	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.
	OnDragDrop (see page 3652)	Occurs when the user drops an object being dragged.
	OnDragOver (see page 3655)	Occurs when the user drags an object over a control.
	OnEndDock (see page 3658)	Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.

	OnEndDrag (see page 3658)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.
	OnEnter (see page 3659)	Occurs when a control receives the input focus.
	OnExit (see page 3662)	Occurs when the input focus shifts away from one control to another.
	OnKeyDown (see page 3666)	Occurs when a user presses any key while the control has focus.
	OnKeyPress (see page 3669)	Occurs when a key is pressed.
	OnKeyUp (see page 3670)	Occurs when the user releases a key that has been pressed.
	OnMouseActivate (see page 3672)	Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.
	OnMouseDown (see page 3673)	Occurs when the user presses a mouse button with the mouse pointer over a control.
	OnMouseEnter (see page 3675)	Occurs when the user moves the mouse into a control.
	OnMouseLeave (see page 3676)	Occurs when the user moves the mouse outside of a control.
	OnMouseMove (see page 3676)	Occurs when the user moves the mouse pointer while the mouse pointer is over a control.
	OnMouseUp (see page 3678)	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.
	OnStartDock (see page 3680)	Occurs when the user begins to drag a control with a DragKind of dkDock.
	OnStartDrag (see page 3681)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

Published Properties

	Name	Description
	Alignment (see page 3631)	Determines how the text is aligned within the text edit control.
	AutoSelect (see page 3631)	Determines whether all the text in the edit control is automatically selected when the control gets focus.
	CharCase (see page 3632)	Determines the case of the text within the edit control.
	Font (see page 3632)	Controls the attributes of text written on or in the control.
	HideSelection (see page 3633)	Determines whether the visual indication of the selected text remains when focus shifts to another control.
	ImeMode (see page 3634)	Determines the behavior of the input method editor (IME).
	ImeName (see page 3634)	Specifies the input method editor (IME) to use for converting keyboard input to Asian language characters.
	Items (see page 3635)	This is Items, a member of class TRibbonComboBox.
	MaxLength (see page 3635)	Specifies the maximum number of characters the user can enter into the edit control.
	NumbersOnly (see page 3636)	Allows only numbers to be typed into the text edit.
	ParentFont (see page 3636)	Determines where a control looks for its font information.
	PasswordChar (see page 3636)	Indicates the character, if any, to display in place of the actual characters typed in the control.
	PopupMenu (see page 3637)	Identifies the pop-up menu associated with the control.
	ReadOnly (see page 3642)	Determines whether the user can change the text of the edit control.
	ShowHint (see page 3642)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	TabOrder (see page 3644)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 3644)	Determines if the user can tab to a control.
	Text (see page 3645)	Contains a text string associated with the control.
	TextHint (see page 3646)	A hint or message to be displayed when the Text property is empty.
	Visible (see page 3646)	Determines whether the component appears onscreen.

63.1.12.2 TRibbonComboBox Properties

The properties of the TRibbonComboBox class are listed here.

Published Properties

	Name	Description
	Alignment (see page 3631)	Determines how the text is aligned within the text edit control.

	AutoSelect (see page 3631)	Determines whether all the text in the edit control is automatically selected when the control gets focus.
	CharCase (see page 3632)	Determines the case of the text within the edit control.
	Font (see page 3632)	Controls the attributes of text written on or in the control.
	HideSelection (see page 3633)	Determines whether the visual indication of the selected text remains when focus shifts to another control.
	ImeMode (see page 3634)	Determines the behavior of the input method editor (IME).
	ImeName (see page 3634)	Specifies the input method editor (IME) to use for converting keyboard input to Asian language characters.
	Items (see page 3635)	This is Items, a member of class TRibbonComboBox.
	MaxLength (see page 3635)	Specifies the maximum number of characters the user can enter into the edit control.
	NumbersOnly (see page 3636)	Allows only numbers to be typed into the text edit.
	ParentFont (see page 3636)	Determines where a control looks for its font information.
	PasswordChar (see page 3636)	Indicates the character, if any, to display in place of the actual characters typed in the control.
	PopupMenu (see page 3637)	Identifies the pop-up menu associated with the control.
	ReadOnly (see page 3642)	Determines whether the user can change the text of the edit control.
	ShowHint (see page 3642)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	TabOrder (see page 3644)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 3644)	Determines if the user can tab to a control.
	Text (see page 3645)	Contains a text string associated with the control.
	TextHint (see page 3646)	A hint or message to be displayed when the Text property is empty.
	Visible (see page 3646)	Determines whether the component appears onscreen.

63.1.12.2.1 TRibbonComboBox.Alignment Property

Determines how the text is aligned within the text edit control.

Pascal

```
property Alignment;
```

C++

```
__property Alignment;
```

Description

Use Alignment to change the way the text is formatted in the text edit control. Alignment can take one of the following values:

Value	Meaning
taLeftJustify	Align text to the left side of the control
taCenter	Center text horizontally in the control.
taRightJustify	Align text to the right side of the control.

See Also

SetAlignment

Text

Displaying and Editing Fields in an Edit Box

63.1.12.2.2 TRibbonComboBox.AutoSelect Property

Determines whether all the text in the edit control is automatically selected when the control gets focus.

Pascal

```
property AutoSelect;
```

C++

```
__property AutoSelect;
```

Description

Set AutoSelect to select all the text when the edit control gets focus. AutoSelect only applies to single-line edit controls.

Use AutoSelect when the user is more likely to replace the text in the edit control than to append to it.

See Also

SelectAll

SelLength

SelStart

SelText

SetSelTextBuf

63.1.12.2.3 TRibbonComboBox.CharCase Property

Determines the case of the text within the edit control.

Pascal

```
property CharCase;
```

C++

```
__property CharCase;
```

Description

Use CharCase to force the contents of the edit control to assume a particular case.

When CharCase is set to ecLowerCase or ecUpperCase, the case of characters is converted as the user types them into the edit control. Changing the CharCase property to ecLowerCase or ecUpperCase changes the actual contents of the text, not just the appearance. Any case information is lost and can't be recaptured by changing CharCase to ecNormal.

See Also

Text

63.1.12.2.4 TRibbonComboBox.Font Property

Controls the attributes of text written on or in the control.

Pascal

```
property Font;
```

C++

```
__property Font;
```

Description

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

C++ Examples:

```
/*
To use this example, place a TComboBox and TRichEdit or
TMemo on a form.  During form creation, a list of font names
is loaded into the combo box. When the combo box is clicked,
the memo or Rich Edit control font is set to the
corresponding font name in the combo box.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < Screen->Fonts->Count; i++)
   ComboBox1->Items->Add(Screen->Fonts->Strings[i]);
//  ComboBox1->Items = Screen->Fonts;  // Another way to do it.
}

void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
  RichEdit1->Font->Name = ComboBox1->Text;
}
```

Delphi Examples:

```
{
To use this example, place a TComboBox and TRichEdit or
TMemo on a form.  During form creation, a list of font names
is loaded into the combo box. When the combo box is clicked,
the memo or Rich Edit control font is set to the
corresponding font name in the combo box.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  i : Integer;
begin
  for i := 0 to Screen.Fonts.Count - 1 do
   ComboBox1.Items.Add(Screen.Fonts.Strings[i]);
//  ComboBox1.Items := Screen.Fonts;  // Another way to do it.
  RichEdit1.Lines.Text := 'How is it to be you?';
end;

procedure TForm1.ComboBox1Click(Sender: TObject);
begin
  RichEdit1.Font.Name := ComboBox1.Text;
end;
```

See Also

DesktopFont

ParentFont

Displaying and Editing Text in a Memo Control

63.1.12.2.5 TRibbonComboBox.HideSelection Property

Determines whether the visual indication of the selected text remains when focus shifts to another control.

Pascal

```
property HideSelection;
```

C++

```
__property HideSelection;
```

Description

Set HideSelection to false to provide visual feedback of the selected portion of the text even when the edit control does not have

focus. Set HideSelection to true to show the selection only when the edit control has focus. HideSelection does not affect the actual value of the selection, only the visual indication. Always setting HideSelection to false can make forms with many edit controls look too busy.

See Also

SelLength

SelStart

SelText

SetSelTextBuf

63.1.12.2.6 TRibbonComboBox.ImeMode Property

Determines the behavior of the input method editor (IME).

Pascal

```
property ImeMode;
```

C++

```
__property ImeMode;
```

Description

Set ImeMode to configure the way an IME processes user keystrokes. An IME is a front-end input processor for Asian language characters. The IME hooks all keyboard input, converts it to Asian characters in a conversion window, and sends the converted characters or strings on to the application.

ImeMode allows a control to influence the type of conversion performed by the IME so that it is appropriate for the input expected by the control. For example, a control that only accepts numeric input might specify an ImeMode of imClose, as no conversion is necessary for numeric input.

Note: The value of ImeMode only takes effect when the control receives focus. To change the value of ImeMode when the control already has input focus, call the SetIme method.

See Also

ImeName

ResetIme

ResetImeComposition

SetIme

SetImeCompositionWindow

63.1.12.2.7 TRibbonComboBox.ImeName Property

Specifies the input method editor (IME) to use for converting keyboard input to Asian language characters.

Pascal

```
property ImeName;
```

C++

```
__property ImeName;
```

Description

Set ImeName to specify which IME to use for converting keystrokes. An IME is a front-end input processor for Asian language characters. The IME hooks all keyboard input, converts it to Asian characters in a conversion window, and sends the converted characters or strings on to the application.

ImeName must specify one of the IMEs that has been installed through the Windows control panel. The property inspector provides a drop-down list of all currently installed IMEs on the system. At runtime, applications can obtain a list of currently installed IMEs from the global Screen variable.

If ImeName specifies an unavailable IME, the IME that was active when the application started is used instead. No exception is generated.

Note: The value of ImeName only takes effect when the control receives focus. To change the value of ImeName after the control has input focus, call the SetIme method.

See Also

ImeMode

ResetIme

SetIme

DefaultIme

Imes

63.1.12.2.8 TRibbonComboBox.Items Property

Pascal

```
property Items: TStrings;
```

C++

```
__property TStrings Items;
```

Description

This is Items, a member of class TRibbonComboBox.

63.1.12.2.9 TRibbonComboBox.MaxLength Property

Specifies the maximum number of characters the user can enter into the edit control.

Pascal

```
property MaxLength;
```

C++

```
__property MaxLength;
```

Description

Use MaxLength to limit the number of characters that can be entered into the edit control. A value of 0 indicates that there is no application-defined limit on the length.

Use MaxLength to limit the length of the text in an edit control if that text will be copied into a fixed-length buffer.

Note: Setting MaxLength will not truncate the existing text, it merely prevents the user from adding more text after reaching the limit of MaxLength characters.

Note: Even when MaxLength is 0, there may be limitations imposed by the operating system on the number of characters that may be entered into an edit control.

Note: The unit of MaxLength is characters (not bytes) if the application is using themes.

See Also

Text

Displaying and Editing Text in a Memo Control

Displaying and Editing Text in a Rich Edit Memo Control

63.1.12.2.10 TRibbonComboBox.NumbersOnly Property

Allows only numbers to be typed into the text edit.

Pascal

```
property NumbersOnly;
```

C++

```
__property NumbersOnly;
```

Description

Use NumbersOnly to prohibit entry of non-numeric characters in the textfield. Note, however, that a user can paste non-numeric characters in the textfield even when this property is set.

63.1.12.2.11 TRibbonComboBox.ParentFont Property

Determines where a control looks for its font information.

Pascal

```
property ParentFont;
```

C++

```
__property ParentFont;
```

Description

To have a control use the same font as its parent control, set ParentFont to true. If ParentFont is false, the control uses its own Font property. Many controls default ParentFont to true so that all the controls in a form or other container present a uniform appearance. When the value of a control's Font property changes, ParentFont becomes false automatically.

When ParentFont is true for a form, the form uses the default font.

See Also

Font

Parent

ParentColor

ParentShowHint

63.1.12.2.12 TRibbonComboBox.PasswordChar Property

Indicates the character, if any, to display in place of the actual characters typed in the control.

Pascal

```
property PasswordChar;
```

C++

```
__property PasswordChar;
```

Description

Use the PasswordChar property to create an edit control that displays a special character in place of any entered text. If PasswordChar is set to the null character (ANSI character zero), the edit control displays its text normally. If PasswordChar is any other character, the edit control displays PasswordChar in place of each character typed. PasswordChar affects the appearance of the edit control only. The value of the Text property reflects the actual characters that are typed.

See Also

Text

63.1.12.2.13 TRibbonComboBox.PopupMenu Property

Identifies the pop-up menu associated with the control.

Pascal

```
property PopupMenu;
```

C++

```
__property PopupMenu;
```

Description

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the control and clicks the right mouse button. If the TPopupMenu's AutoPopup property is true, the pop-up menu appears automatically. If the menu's AutoPopup property is false, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

C++ Examples:

```
/*
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  PopupMenu1->AutoPopup = true;
  Edit1->PopupMenu = PopupMenu1;
  Edit2->PopupMenu = PopupMenu1;
}

/*
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
*/
void __fastcall TForm1::Copy1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
```

```
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CopyToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Cut1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CutToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Paste1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->PasteFromClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}
/*
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
    {
        switch (ClickedItem->Tag)
        {
```

```
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
            }
        }
    }
}

/*
To edit or destroy an item, grab its pointer via the Items
property.
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}

void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  PopupMenu1.AutoPopup := True;
```

```
    Edit1.PopupMenu := PopupMenu1;
    Edit2.PopupMenu := PopupMenu1;
  end;

  {
  These are the cut, copy, and paste OnClick events for the commands
  on the pop-up menu. Double click on the pop-up menu and add three
  TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
  The names will be automatically set to "Cut1", "Copy1", and
  "Paste1".  Then double click on the OnClick events and they will
  be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
  Note: only the selected portion of the string in the edit box will
  be copied, cut or replaced.
  }
  procedure TForm1.Copy1Click(Sender: TObject);
  begin
    if PopupMenu1.PopupComponent = Edit1 then
     Edit1.CopyToClipboard
    else if PopupMenu1.PopupComponent = Edit2 then
      Edit2.CopyToClipboard
    else
      Beep;
  end;

  procedure TForm1.Cut1Click(Sender: TObject);
  begin
    if PopupMenu1.PopupComponent = Edit1 then
      Edit1.CutToClipboard
    else if PopupMenu1.PopupComponent = Edit2 then
      Edit2.CutToClipboard
    else
      Beep;
  end;

  procedure TForm1.Paste1Click(Sender: TObject);
  begin
    if PopupMenu1.PopupComponent = Edit1 then
      Edit1.PasteFromClipboard
    else if PopupMenu1.PopupComponent = Edit2 then
      Edit2.PasteFromClipboard
    else
      Beep;
  end;
  {
  This example demonstrates how to add and delete menu items
  to a popup menu at runtime and assign an event handler to
  the OnClick event.  Place a TPopupMenu and three buttons on
  the form named "AddButton", "EditButton",  and
  "DestroyButton" and add OnClick events to all three buttons.
  Put the TPopupMenu in the PopupMenu property of the form.
  Place the PopupMenuItemsClick procedure in the TForm1 type
  declaration so that it can be used as the method call for
  the menu item OnClick event.
  }

  type
    TForm1 = class(TForm)
      AddButton: TButton;
      EditButton: TButton;
      DestroyButton: TButton;
      PopupMenu1: TPopupMenu;
      procedure AddButtonClick(Sender: TObject);
      procedure EditButtonClick(Sender: TObject);
      procedure DestroyButtonClick(Sender: TObject);
      procedure FormCreate(Sender: TObject);
    private
```

```
    { Private declarations }
  public
    procedure PopupMenuItemsClick(Sender: TObject);

  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.AddButtonClick(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
    // The owner (PopupMenu1) will cleanup this menu item.
  NewItem := TMenuItem.Create(PopupMenu1); // create the new item
  index := PopupMenu1.Items.Count;
  PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
  NewItem.Caption := 'Menu Item ' + IntToStr(index);
  NewItem.Tag := index;
  NewItem.OnClick :=
    PopupMenuItemsClick; // assign it an event handler
end;

procedure TForm1.PopupMenuItemsClick(Sender: TObject);
begin
  with Sender as TMenuItem do
  begin
    case Tag of
      0:  ShowMessage('first item clicked');
      1:  ShowMessage('second item clicked');
      2:  ShowMessage('third item clicked');
      3:  ShowMessage('fourth item clicked');
    end;
  end;
end;

{
To edit or destroy an item, grab its pointer via the Items
property.
}
procedure TForm1.EditButtonClick(Sender: TObject);
var
  ItemToEdit: TMenuItem;
begin
  ItemToEdit := PopupMenu.Items[1];
  ItemToEdit.Caption := 'Changed Caption';
end;

procedure TForm1.DestroyButtonClick(Sender: TObject);
var
  ItemToDelete: TMenuItem;
begin
  ItemToDelete := PopupMenu.Items[2];
  ItemToDelete.Free;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
```

```
    for index := 0 to 3 do
    begin
        // The owner (PopupMenu1) will cleanup this menu item.
      NewItem := TMenuItem.Create(PopupMenu1); // create the new item
      PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
      NewItem.Caption := 'Menu Item ' + IntToStr(index);
      NewItem.Tag := index;
      NewItem.OnClick :=
        PopupMenuItemsClick; // assign it an event handler
    end;
  end;
```

See Also

AutoPopup (see page 1437)

Popup (see page 1434)

OnContextPopup

63.1.12.2.14 TRibbonComboBox.ReadOnly Property

Determines whether the user can change the text of the edit control.

Pascal

```
property ReadOnly;
```

C++

```
__property ReadOnly;
```

Description

To restrict the edit control to display only, set the ReadOnly property to true. Set ReadOnly to false to allow the contents of the edit control to be edited.

Setting ReadOnly to true ensures that the text is not altered, while still allowing the user to select text. The selected text can then be manipulated by the application, or copied to the Clipboard.

See Also

CopyToClipboard

GetSelTextBuf

63.1.12.2.15 TRibbonComboBox.ShowHint Property

Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.

Pascal

```
property ShowHint;
```

C++

```
__property ShowHint;
```

Description

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either

the control's own ShowHint property must be true, or

the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

C++ Examples:

```
/*
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  Edit1->Hint = "Enter your name";
  Edit1->ShowHint = true;

  char string[10];
  char index[3];
  for(int i = 1; i <= 10; i++)
  {
    strcpy(string, "Item");
    itoa(i, index, 10);
    strcat(string, index);
    ListBox1->Items->Add(string);
  }
  ListBox1->Hint = "Select an item";
  ListBox1->ShowHint = true;
  Application->ShowHint = true;
}
/*
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
*/
```

Delphi Examples:

```
{
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
  Edit1.Hint := 'Enter your name';
  Edit1.ShowHint := True;
  with ListBox1 do
  begin
    for I := 1 to 10 do
      Items.Add('Item ' + IntToStr(I));
    Hint := 'Select an item';
    ShowHint := True;
  end;
  Application.ShowHint := True;
end;
{
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
}
```

See Also

Hint

ParentShowHint

ShowHint

Displaying Fly-over Help

63.1.12.2.16 TRibbonComboBox.TabOrder Property

Indicates the position of the control in its parent's tab order.

Pascal

```
property TabOrder;
```

C++

```
__property TabOrder;
```

Description

TabOrder is the order in which child windows are visited when the user presses the Tab key. The control with the TabOrder value of 0 is the control that has the focus when the form first appears.

Initially, the tab order is always the order in which the controls were added to the form. The first control added to the form has a TabOrder value of 0, the second is 1, the third is 2, and so on. Change this by changing the TabOrder property.

Each control has a unique tab-order value within its parent. If you change the TabOrder property value of one control to be the same as the value of a second control, the TabOrder value for all the other controls changes. For example, suppose a control is sixth in the tab order. If you change the control's TabOrder property value to 3 (making the control fourth in the tab order), the control that was originally fourth in the tab order now becomes fifth, and the control that was fifth becomes sixth.

Assigning TabOrder a value greater than the number of controls contained in the parent control moves the control to the end of the tab order. The control does not take on the assigned value of TabOrder, but instead is given the number that assures the control is the last in the tab order.

Note: TabOrder is meaningful only if the TabStop property is true and if the control has a parent. (The TabOrder property of a form is not used unless the form is the child of another form.) A control with a TabOrder of -1 has no parent, and therefore cannot be reached by pressing the Tab key. To remove a parented control from the Tab order, set its TabStop property to false.

See Also

FindNextControl

TabStop

63.1.12.2.17 TRibbonComboBox.TabStop Property

Determines if the user can tab to a control.

Pascal

```
property TabStop;
```

C++

```
__property TabStop;
```

Description

Use the TabStop to allow or disallow access to the control using the Tab key.

If TabStop is true, the control is in the tab order. If TabStop is false, the control is not in the tab order and the user can't press the Tab key to move to the control.

Note: TabStop is not meaningful for a form unless the form assigns another form to be its parent.

See Also

TabOrder

63.1.12.2.18 TRibbonComboBox.Text Property

Contains a text string associated with the control.

Pascal

```
property Text;
```

C++

```
__property Text;
```

Description

Use the Text property to read the Text of the control or specify a new string for the Text value. By default, Text is the control name. For edit controls and memos, the Text appears within the control. For combo boxes, the Text is the content of the edit control portion of the combo box.

Note: Controls that display text use either the Caption property or the Text property to specify the text value. Which property is used depends on the type of control. In general, Caption is used for text that appears as a window title or label, while Text is used for text that appears as the content of a control.

C++ Examples:

```
/*
This example requires two edit boxes and a button.  This example
copies the text in an edit box into a null-terminated string, and
puts this string in another edit box when the user clicks the
button on the form.
*/
#include <memory>       //for STL auto_ptr class

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  int Size = Edit1->GetTextLen(); //Get length of string in Edit1
  Size++;                       //Add room for null character
//  Char *Buffer = new Char[Size];  //Creates Buffer dynamic variable
  std::auto_ptr<Char> Buffer(new Char[Size]);
  Edit1->GetTextBuf(Buffer.get(),Size); //Puts Edit1->Text into Buffer
  Edit2->Text = Buffer.get();
}
/*
Note: the same thing could be accomplished more simply as follows:
*/
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  Edit2->Text = Edit1->Text;
}
```

Delphi Examples:

```
{
This example requires two edit boxes and a button.  This example
copies the text in an edit box into a null-terminated string, and
puts this string in another edit box when the user clicks the
button on the form.
}
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Buffer: PChar;
  Size: Byte;
begin
  Size := Edit1.GetTextLen;       {Get length of string in Edit1}
  Inc(Size);                      {Add room for null character}
  GetMem(Buffer, Size);           {Creates Buffer dynamic variable}
  Edit1.GetTextBuf(Buffer,Size);  {Puts Edit1.Text into Buffer}
  Edit2.Text := StrPas(Buffer);   {Converts Buffer to a Pascal-style string}
  FreeMem(Buffer, Size);{Frees memory allocated to Buffer}
end;

{
Note: the same thing could be accomplished more simply as follows:
}
procedure TForm1.Button2Click(Sender: TObject);
begin
  Edit2.Text := Edit1.Text;
end;
```

See Also

Caption

Lines

63.1.12.2.19 TRibbonComboBox.TextHint Property

A hint or message to be displayed when the Text property is empty.

Pascal

```
property TextHint;
```

C++

```
__property TextHint;
```

Description

Use TextHint to cue the user on what kind of entry is expected in the text field. Runtime themes must be enabled.

63.1.12.2.20 TRibbonComboBox.Visible Property

Determines whether the component appears onscreen.

Pascal

```
property Visible;
```

C++

```
__property Visible;
```

Description

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

Calling the Show method sets the control's Visible property to true. Calling the Hide method sets it to false.

C++ Examples:

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
```

```
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
    pPage->PageControl = PageControl1;
    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this); // The owner will clean this up.
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
    begin
      PageControl := PageControl1;
      Caption := 'TabSheet' + IntToStr(i);
      with TEdit.Create(Self) do
      begin
        Parent := PageControl1.Pages[i];
        Left := Random(PageControl1.ActivePage.ClientWidth - Width);
        Top := Random(PageControl1.ActivePage.ClientHeight - Height);
      end;
    end;
    PageControl1Change(Sender);
end;
```

See Also

Enabled

Hide

Show

VisibleChanging

63.1.12.3 TRibbonComboBox Events

The events of the TRibbonComboBox class are listed here.

Published Events

	Name	Description
	OnChange (see page 3648)	Occurs when the text for the edit control may have changed.
	OnClick (see page 3649)	Occurs when the user clicks the control.
	OnContextPopup (see page 3651)	Occurs when the user right-clicks the control or otherwise invokes the popup menu (such as using the keyboard).
	OnDblClick (see page 3651)	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.
	OnDragDrop (see page 3652)	Occurs when the user drops an object being dragged.
	OnDragOver (see page 3655)	Occurs when the user drags an object over a control.
	OnEndDock (see page 3658)	Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.
	OnEndDrag (see page 3658)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.
	OnEnter (see page 3659)	Occurs when a control receives the input focus.
	OnExit (see page 3662)	Occurs when the input focus shifts away from one control to another.
	OnKeyDown (see page 3666)	Occurs when a user presses any key while the control has focus.
	OnKeyPress (see page 3669)	Occurs when a key is pressed.
	OnKeyUp (see page 3670)	Occurs when the user releases a key that has been pressed.
	OnMouseActivate (see page 3672)	Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.
	OnMouseDown (see page 3673)	Occurs when the user presses a mouse button with the mouse pointer over a control.
	OnMouseEnter (see page 3675)	Occurs when the user moves the mouse into a control.
	OnMouseLeave (see page 3676)	Occurs when the user moves the mouse outside of a control.
	OnMouseMove (see page 3676)	Occurs when the user moves the mouse pointer while the mouse pointer is over a control.
	OnMouseUp (see page 3678)	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.
	OnStartDock (see page 3680)	Occurs when the user begins to drag a control with a DragKind of dkDock.
	OnStartDrag (see page 3681)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

63.1.12.3.1 TRibbonComboBox.OnChange Event

Occurs when the text for the edit control may have changed.

Pascal

```
property OnChange;
```

C++

```
__property OnChange;
```

Description

Write an OnChange event handler to take specific action whenever the text for the edit control may have changed. Use the Modified property to see if a change actually occurred. The Text property of the edit control will already be updated to reflect any changes. This event provides the first opportunity to respond to modifications that the user types into the edit control.

See Also

Change

Text

63.1.12.3.2 TRibbonComboBox.OnClick Event

Occurs when the user clicks the control.

Pascal

```
property OnClick;
```

C++

```
__property OnClick;
```

Description

Use the OnClick event handler to respond when the user clicks the control. If the control has an associated action, and that action has an OnExecute method, the action's OnExecute method responds to click events unless it is superseded by an OnClick event handler.

Usually OnClick occurs when the user presses and releases the left mouse button with the mouse pointer over the control. This event can also occur when:

- The user selects an item in a grid, outline, list, or combo box by pressing an arrow key.
- The user presses Spacebar while a button or check box has focus.
- The user presses Enter when the active form has a default button (specified by the Default property).
- The user presses Esc when the active form has a cancel button (specified by the Cancel property).
- The user presses the accelerator key for a button or check box. For example, if the value of the Caption property of a check box is '&Bold', the B is underlined at runtime and the OnClick event of the check box is triggered when the user presses Alt+B. However, focus does not move to the control in these instances.
- The Checked property of a radio button is set to true.
- The value of the Checked property of a check box is changed.
- The Click method of a menu item is called.

For a form, an OnClick event occurs when the user clicks a blank area of the form or on a disabled component.

C++ Examples:

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}
```

```
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
    pPage->PageControl = PageControl1;
    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this); // The owner will clean this up.
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
    begin
      PageControl := PageControl1;
      Caption := 'TabSheet' + IntToStr(i);
      with TEdit.Create(Self) do
      begin
        Parent := PageControl1.Pages[i];
        Left := Random(PageControl1.ActivePage.ClientWidth - Width);
        Top := Random(PageControl1.ActivePage.ClientHeight - Height);
      end;
    end;
    PageControl1Change(Sender);
end;
```

See Also

OnDblClick

Cancel

Default

Click (see page 1375)

OnExecute

Action

Adding Silent Video Clips to an Application

63.1.12.3.3 TRibbonComboBox.OnContextPopup Event

Occurs when the user right-clicks the control or otherwise invokes the popup menu (such as using the keyboard).

Pascal

```
property OnContextPopup;
```

C++

```
__property OnContextPopup;
```

Description

The OnContextPopup handler is called when the user uses the mouse or keyboard to request a popup menu. The OnContextPopup event is generated by a WM_CONTEXTMENU message, which is itself generated by the user clicking the right mouse button or by pressing SHIFT+F10 or the Applications key.

This event is especially useful when the control does not have an associated popup menu (the PopupMenu property is not set) or if the AutoPopup property of the control's associated popup menu is false. However, the OnContextPopup can also be used to override the automatic context menu that appears when the control has an associated popup menu with an AutoPopup property of true. In this last case, if the event handler displays its own menu, it should set the Handled parameter to true to suppress the default context menu.

The handler's MousePos parameter indicates the position of the mouse, in client coordinates.. If the event was not generated by a mouse click, MousePos is (-1,-1).

Note: Parent controls receive an OnContextPopup event before their child controls. In addition, for many child controls, the default window procedure causes the parent control to receive an OnContextPopup event after the child control. As a result, when parent controls do not set Handled to true in an OnContextPopup event handler, the event handler may be called multiple times for each context menu invocation.

See Also

TPopupMenu (see page 1430)

PopupMenu

AutoPopup (see page 1437)

63.1.12.3.4 TRibbonComboBox.OnDblClick Event

Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.

Pascal

```
property OnDblClick;
```

C++

```
__property OnDblClick;
```

Description

Use the OnDblClick event to respond to mouse double-clicks.

See Also

OnClick

Responding to User Actions at Runtime

63.1.12.3.5 TRibbonComboBox.OnDragDrop Event

Occurs when the user drops an object being dragged.

Pascal

```
property OnDragDrop;
```

C++

```
__property OnDragDrop;
```

Description

Use the OnDragDrop event handler to specify what happens when the user drops an object. The Source parameter of the OnDragDrop event is the object being dropped, and the Sender is the control the object is being dropped on. The X and Y parameters are the coordinates of the mouse positioned over the control.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}

void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
```

```
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
}
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  TForm1(Source).Left := X;
  TForm1(Source).Top := Y;
end;

procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
```

```
begin
  Accept := Source is TLabel;
end;

procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := False;
  if (Source is TLabel) and (State = dsDragEnter) then
    (Source as TLabel).EndDrag(False);
end;
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
dropped label.
}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Not');
  ListBox1.Items.Add('In');
  ListBox1.Items.Add('Alphabetical');
  ListBox1.Items.Add('Order');
end;

// This OnDragOver event handler permits the list box to
// accept a dropped label:

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

// This OnDragDrop event handler implements the drop behavior.

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
    with Sender as TListBox do
    begin
      Font := (Source as TLabel).Font;
      Color := (Source as TLabel).Color;
      DoubleBuffered := true;
      Color := clWindow;
    end;
  end;
end;
```

See Also

BeginDrag

DragDrop

Dragging

EndDrag

OnDragOver

OnEndDrag

OnStartDrag

TDragObject

63.1.12.3.6 TRibbonComboBox.OnDragOver Event

Occurs when the user drags an object over a control.

Pascal

```
property OnDragOver;
```

C++

```
__property OnDragOver;
```

Description

Use an OnDragOver event to signal that the control can accept a dragged object so the user can drop or dock it.

Within the OnDragOver event handler, change the Accept parameter to false to reject the dragged object. Leave Accept as true to allow the user to drop or dock the dragged object on the control.

To change the shape of the cursor, indicating that the control can accept the dragged object, change the value of the DragCursor property for the control before the OnDragOver event occurs.

The Source is the object being dragged, the Sender is the potential drop or dock site, and X and Y are screen coordinates in pixels. The State parameter specifies how the dragged object is moving over the control.

Note: Within the OnDragOver event handler, the Accept parameter defaults to true. However, if an OnDragOver event handler is not supplied, the control rejects the dragged object, as if the Accept parameter were changed to false.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}

void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
```

```
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
}
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
```

```
{$R *.dfm}

procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  TForm1(Source).Left := X;
  TForm1(Source).Top := Y;
end;

procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := False;
  if (Source is TLabel) and (State = dsDragEnter) then
    (Source as TLabel).EndDrag(False);
end;
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
dropped label.
}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Not');
  ListBox1.Items.Add('In');
  ListBox1.Items.Add('Alphabetical');
  ListBox1.Items.Add('Order');
end;

// This OnDragOver event handler permits the list box to
// accept a dropped label:

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

// This OnDragDrop event handler implements the drop behavior.

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
    with Sender as TListBox do
    begin
      Font := (Source as TLabel).Font;
      Color := (Source as TLabel).Color;
      DoubleBuffered := true;
      Color := clWindow;
    end;
  end;
end;
```

See Also

DragCursor

OnDragDrop

OnEndDrag

OnStartDrag

TDragObject

DragKind

OnEndDock

OnDockDrop

OnStartDock

63.1.12.3.7 TRibbonComboBox.OnEndDock Event

Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.

Pascal

```
property OnEndDock;
```

C++

```
__property OnEndDock;
```

Description

Use OnEndDock to specify actions or special processing that when a drag-and-dock operation stops.

See Also

DragKind

OnDragOver

OnDockDrop

OnStartDock

63.1.12.3.8 TRibbonComboBox.OnEndDrag Event

Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.

Pascal

```
property OnEndDrag;
```

C++

```
__property OnEndDrag;
```

Description

Use the OnEndDrag event handler to specify any special processing that occurs when dragging stops.

C++ Examples:

```
/*
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
*/
void __fastcall TForm1::Label1EndDrag(TObject *Sender, TObject *Target, int X,
      int Y)
```

```
{
  if (Sender->ClassNameIs("TLabel"))
  {
    AnsiString S = (dynamic_cast<TLabel *>(Sender))->Name + " was dropped and ";
    if (Target)
      S = S + "accepted!";
    else
      S = S + "rejected.";
    Status->Caption = S;
  }
}
```

Delphi Examples:

```
{
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
}
procedure TForm1.Label1EndDrag(Sender, Target: TObject; X, Y: Integer);
var
  S: string;
begin
  S := (Sender as TLabel).Name + ' was dropped... and ';
  if Target <> nil then
    S := S + 'accepted!'
  else
    S := S + 'rejected!';
  Status.Caption := S;
end;
```

See Also

OnDragDrop

OnDragOver

OnStartDrag

DragKind

63.1.12.3.9 TRibbonComboBox.OnEnter Event

Occurs when a control receives the input focus.

Pascal

```
property OnEnter;
```

C++

```
__property OnEnter;
```

Description

Use the OnEnter event handler to cause any special processing to occur when a control becomes active.

The OnEnter event does not occur when switching between forms or between another application and the application that includes the control.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnEnter event occurs for the container before the OnEnter event of the contained control.

Similarly, an OnExit event of the container occurs after the OnExit event of the control in a container when focus moves to another control outside the container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

C++ Examples:

```
/*
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
*/
TWinControl *myActiveControl;

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  FontDialog1->Options << fdApplyButton;
  FontDialog1->Execute();
}

void __fastcall TForm1::FontDialog1Apply(TObject *Sender, HWND Wnd)
{
  if (myActiveControl->ClassNameIs("TEdit"))
   (dynamic_cast<TEdit *>(myActiveControl))->Font->Assign(FontDialog1->Font);
  else if (myActiveControl->ClassNameIs("TRichEdit"))
    (dynamic_cast<TRichEdit *>(myActiveControl))->SelAttributes->Assign(FontDialog1->Font);
  else
    MessageBeep(0);
}

void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  myActiveControl = ActiveControl;
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
```

```
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
   FontDialog1.Options := FontDialog1.Options + [fdApplyButton];
   FontDialog1.Execute;
end;

procedure TForm1.Edit1Enter(Sender: TObject);
begin
  myActiveControl := ActiveControl;
end;

procedure TForm1.FontDialog1Apply(Sender: TObject; Wnd: HWND);
begin
  if myActiveControl is TEdit then
    with myActiveControl as TEdit do
       Font.Assign(TFontDialog(Sender).Font)
  else if myActiveControl is TRichEdit then
    with myActiveControl as TRichEdit do
      SelAttributes.Assign(TFontDialog(Sender).Font)
  else
    Beep;
end;
{
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
}
procedure TForm1.Edit1Enter(Sender: TObject);
begin
  Edit1.Color := clYellow;
end;

procedure TForm1.Edit1Exit(Sender: TObject);
begin
  Edit1.Color := clWindow;
end;

procedure TForm1.Memo1Enter(Sender: TObject);
begin
  Memo1.Color := clYellow;
end;

procedure TForm1.Memo1Exit(Sender: TObject);
begin
  Memo1.Color := clWindow;
end;
```

See Also

OnExit

ActiveControl

OnActivate

Using a Single Navigator for Multiple Datasets

63.1.12.3.10 TRibbonComboBox.OnExit Event

Occurs when the input focus shifts away from one control to another.

Pascal

```
property OnExit;
```

C++

```
__property OnExit;
```

Description

Use the OnExit event handler to provide special processing when the control ceases to be active.

The OnExit event does not occur when switching between forms or between another application and your application.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnExit event occurs for the control inside the container before the OnExit event of the container.

Similarly, an OnEnter event of the container occurs before the OnEnter event of the control in a container when focus moves to a control inside a container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

Note: In some control classes the ActiveControl property updates before the OnExit event occurs.

C++ Examples:

```
/*
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255 ,
as this validation test cannot be accomplished
using the edit mask property alone.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  // initialize the maskedit boxes to contain the empty string
  MaskEdit1->Text = "";
```

```
  MaskEdit2->Text = "";
  MaskEdit3->Text = "";

  // use the Courier font in the maskedit boxes
  MaskEdit1->Font->Name = "Courier";
  MaskEdit2->Font->Name = "Courier";
  MaskEdit3->Font->Name = "Courier";

  // initialize the mask of each maskedit box
  // according to IP address coding conventions
  MaskEdit1->EditMask = "!099.099.099.099;1; ";
  MaskEdit2->EditMask = "!099.099.099.099;1; ";
  MaskEdit3->EditMask = "!099.099.099.099;1; ";

  MaskEdit2->OnExit = MaskEdit2Exit;
}

void __fastcall TForm1::MaskEdit2Exit(TObject *Sender)
{
  // extract the net and host address from the IP
  String IP = MaskEdit1->Text;
  int net1 = IP.SubString(0, 3).TrimRight().ToInt();
  int net2 = IP.SubString(5, 3).TrimRight().ToInt();
  int host1 = IP.SubString(9, 3).TrimRight().ToInt();
  int host2 = IP.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (net1 < 0 || net1 > 255 ||
      net2 < 0 || net2 > 255 ||
      host1 < 0 || host1 > 255 ||
      host2 < 0 || host2 > 255)
    throw(Exception("Not a valid IP address."));

  // extract the net and host mask from the subnet mask
  String mask = MaskEdit2->Text;
  int netmask1 = mask.SubString(0, 3).TrimRight().ToInt();
  int netmask2 = mask.SubString(5, 3).TrimRight().ToInt();
  int hostmask1 = mask.SubString(9, 3).TrimRight().ToInt();
  int hostmask2 = mask.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (netmask1 < 0 || netmask1 > 255 ||
      netmask2 < 0 || netmask2 > 255 ||
      hostmask1 < 0 || hostmask1 > 255 ||
      hostmask2 < 0 || hostmask2 > 255)
    throw(Exception("Not a valid subnet mask."));

  // compute the subnet address using bit-wise AND
  int sub_net1 = net1 & netmask1;
  int sub_net2 = net2 & netmask2;
  int sub_host1 = host1 & hostmask1;
  int sub_host2 = host2 & hostmask2;

  // display the subnet address
  MaskEdit3->Text =
    String(sub_net1) + "." + String(sub_net2) + "." +
    String(sub_host1) + "." + String(sub_host2);
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
```

```
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255,
as this validation test cannot be accomplished
using the edit mask property alone.
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  { initialize the maskedit boxes to contain the empty string }
  MaskEdit1.Text := '';
  MaskEdit2.Text := '';
  MaskEdit3.Text := '';

  { use the Courier font in the maskedit boxes }
  MaskEdit1.Font.Name := 'Courier';
  MaskEdit2.Font.Name := 'Courier';
  MaskEdit3.Font.Name := 'Courier';

  {
  initialize the mask of each maskedit box
  according to IP address coding conventions
  }
  MaskEdit1.EditMask := '!099.099.099.099;1; ';
  MaskEdit2.EditMask := '!099.099.099.099;1; ';
  MaskEdit3.EditMask := '!099.099.099.099;1; ';

  MaskEdit2.OnExit := MaskEdit2Exit;
end;

procedure TForm1.MaskEdit2Exit(Sender: TObject);
var
  net1, net2, host1, host2,
  netmask1, netmask2, hostmask1, hostmask2,
  sub_net1, sub_net2, sub_host1, sub_host2: Integer;

  IP, mask: String;
```

```
begin
  // extract the net and host address from the IP
  IP := MaskEdit1.Text;
  net1 := StrToInt(TrimRight(Copy(IP, 0, 3)));
  net2 := StrToInt(TrimRight(Copy(IP, 5, 3)));
  host1 := StrToInt(TrimRight(Copy(IP, 9, 3)));
  host2 := StrToInt(TrimRight(Copy(IP, 13, 3)));

  // a range test that you cannot validate through edit masks
  if ((net1 < 0) Or (net1 > 255) Or
      (net2 < 0) Or (net2 > 255) Or
      (host1 < 0) Or (host1 > 255) Or
      (host2 < 0) Or (host2 > 255)) then
    raise EArgumentException.Create('Not a valid IP address.');

  // extract the net and host mask from the subnet mask
  mask := MaskEdit2.Text;
  netmask1 := StrToInt(TrimRight(Copy(mask, 0, 3)));
  netmask2 := StrToInt(TrimRight(Copy(mask, 5, 3)));
  hostmask1 := StrToInt(TrimRight(Copy(mask, 9, 3)));
  hostmask2 := StrToInt(TrimRight(Copy(mask, 13, 3)));

  // a range test that you cannot validate through edit masks
  if ((netmask1 < 0) Or (netmask1 > 255) Or
      (netmask2 < 0) Or (netmask2 > 255) Or
      (hostmask1 < 0) Or (hostmask1 > 255) Or
      (hostmask2 < 0) Or (hostmask2 > 255)) then
    raise EArgumentException.Create('Not a valid subnet mask.');

  // compute the subnet address using bit-wise AND
  sub_net1 := net1 And netmask1;
  sub_net2 := net2 And netmask2;
  sub_host1 := host1 And hostmask1;
  sub_host2 := host2 And hostmask2;

  // display the subnet address
  MaskEdit3.Text :=
    IntToStr(sub_net1) + '.' + IntToStr(sub_net2) + '.' +
    IntToStr(sub_host1) + '.' + IntToStr(sub_host2);
end;
{
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
}
procedure TForm1.Edit1Enter(Sender: TObject);
begin
  Edit1.Color := clYellow;
end;

procedure TForm1.Edit1Exit(Sender: TObject);
begin
  Edit1.Color := clWindow;
end;

procedure TForm1.Memo1Enter(Sender: TObject);
begin
  Memo1.Color := clYellow;
end;

procedure TForm1.Memo1Exit(Sender: TObject);
begin
```

```
  Memo1.Color := clWindow;
end;
```

See Also

OnEnter

ActiveControl

63.1.12.3.11 TRibbonComboBox.OnKeyDown Event

Occurs when a user presses any key while the control has focus.

Pascal

```
property OnKeyDown;
```

C++

```
__property OnKeyDown;
```

Description

Use the OnKeyDown event handler to specify special processing to occur when a key is pressed. The OnKeyDown handler can respond to keyboard keys, including function keys and keys combined with the `Shift`, `Alt`, and `Ctrl` keys, and pressed mouse buttons.

Note: TCheckBox does not fire OnKeyDown events for arrow keys.

The TKeyEvent type points to a method that handles keyboard events.

The Key parameter is the key on the keyboard. For non-alphanumeric keys, use virtual key codes to determine the key pressed. For more information, see Virtual Key codes.

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

An application gets Windows WM_KEYDOWN messages for all keys when the user presses a key. These messages indirectly fire the OnKeyDown event. If you set the Key to 0 there you prevent only any further processing of this message. But for keys that generate characters Windows also produces WM_CHAR. At the time your OnKeyDown event fires, the WM_CHAR message for the key will already be in the message queue. Setting Key to 0 does not stop it from being delivered, so it fires the OnKeyPress event. You must set the Key to #0 to prevent the control from intercepting the WM_CHAR message.

This method of organizing key processing has advantages. Code that only deals with characters, including control characters like #13 for carriage return, #3 for Ctrl-C and so on, should go into the OnKeyPress event. Code that deals with keys that do not generate characters should be put into the OnKeyDown event.

C++ Examples:

```
/*
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
*/
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  if (Key == VK_F1)
  {
    FormColor = Form1->Color;
    Form1->Color = clAqua;
    if (ComboBox1->Text == "KeyDown")
```

```
      ComboBox1->Text = "KeyDown2";
    else
      ComboBox1->Text = "KeyDown";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}

void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key, TShiftState Shift)
{
  if (Key == VK_F1)
  {
    Form1->Color = FormColor;
    if (ComboBox1->Text == "KeyUp")
      ComboBox1->Text = "KeyUp2";
    else
      ComboBox1->Text = "KeyUp";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}
/*
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
*/
#include <Printers.hpp>

void __fastcall TForm1::FormCreate(TObject *Sender)
{
   KeyPreview = True;

}

void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)

{
  if (Key == VK_ESCAPE && Printer()->Printing)
  {
    Printer()->Abort();
    MessageDlg("Printing aborted", mtInformation,
               TMsgDlgButtons() << mbOK,0);
  }
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMemo *Memo1 = new TMemo(Form1); // The parent (Form1) will clean up this object.
  Memo1->Parent = Form1;
  Memo1->Visible = True;
  Memo1->Width = 700;
  if (OpenDialog1->Execute())
  {
    Memo1->Lines->LoadFromFile(OpenDialog1->FileName);

    Printer()->BeginDoc();
    int X = 200;
    int Y = 200;
    for (int I = 0; I <= 10; I++)
      if (!Printer()->Aborted)
      {
        Printer()->Canvas->TextOut(X, Y, Memo1->Lines->Strings[I]);
        Y = Y + 80;
        if (Y > (Printer()->PageHeight - 300))
        {
          Y = 200;
```

```
          Printer()->NewPage();
          Sleep(1000);  // to give you time to abort!
        }
      }
    if (!Printer()->Aborted) Printer()->EndDoc();
  }
}
```

Delphi Examples:

```
{
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
}
var
  FormColor: TColor;
procedure TForm1.FormCreate(Sender: TObject);
begin
  KeyPreview := True;
  FormColor := Form1.Color;
end;

procedure TForm1.FormKeyDown(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := clAqua;
    if (ComboBox1.Text = 'KeyDown') then
      ComboBox1.Text := 'KeyDown2'
    else
      ComboBox1.Text := 'KeyDown';
    WriteLn(ComboBox1.Text);
  end;
end;

procedure TForm1.FormKeyUp(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := FormColor;
    if (ComboBox1.Text = 'KeyUp') then
      ComboBox1.Text := 'KeyUp2'
    else
      ComboBox1.Text := 'KeyUp';
    WriteLn(ComboBox1.Text);
  end;
end;
{
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
}

uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  I, X, Y: Integer;
  Memo1 : TMemo;
  r: TRect;
```

```
begin
  Memo1 := TMemo.Create(Form1);
  Memo1.Parent := Form1;
  Memo1.Visible := True;
  Memo1.Width := 700;
  if (OpenDialog1.Execute) then
  begin
    Memo1.Lines.LoadFromFile(OpenDialog1.FileName);

    Printer.BeginDoc;
    X := 200;
    Y := 200;
    for I := 0 to 140 do
      if (not Printer.Aborted) then
      begin
        Printer.Canvas.TextOut(X, Y, Memo1.Lines[I]);
        Y := Y + 80;
        if (Y > (Printer.PageHeight - 300)) then
        begin
          Y := 200;
          Printer.NewPage;
          Sleep(1000);  // to give you time to abort!
        end;
      end;
    if (not Printer.Aborted) then Printer.EndDoc;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
   KeyPreview := True;
end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
if (Key=VK_ESCAPE) and Printer.Printing then
  begin
  Printer.Abort;
  MessageDlg('Printing aborted', mtInformation, [mbOK],0);
  end;
end;
```

See Also

KeyPreview

OnKeyPress

OnKeyUp

63.1.12.3.12 TRibbonComboBox.OnKeyPress Event

Occurs when a key is pressed.

Pascal

```
property OnKeyPress;
```

C++

```
__property OnKeyPress;
```

Description

Use the OnKeyPress event handler to make something happen as a result of a single character key press.

The Key parameter in the OnKeyPress event handler is of type Char; therefore, the OnKeyPress event registers the ASCII

character of the key pressed. Keys that don't correspond to an ASCII Char value (Shift or F1, for example) don't generate an OnKeyPress event. Key combinations (such as Shift+A), generate only one OnKeyPress event (for this example, Shift+A results in a Key value of "A" if Caps Lock is off). To respond to non-ASCII keys or key combinations, use the OnKeyDown or OnKeyUp event handlers.

An application gets Windows WM_KEYDOWN messages for all keys when the user presses a key. These messages indirectly fire the OnKeyDown event. If you set the Key to 0 there you prevent only any further processing of this message. But for keys that generate characters Windows also produces WM_CHAR. At the time your OnKeyDown event fires, the WM_CHAR message for the key will already be in the message queue. Setting Key to 0 does not stop it from being delivered, so it fires the OnKeyPress event. You must set the Key to #0 to prevent the control from intercepting the WM_CHAR message.

This method of organizing key processing has advantages. Code that only deals with characters, including control characters like #13 for carriage return, #3 for Ctrl-C and so on, should go into the OnKeyPress event. Code that deals with keys that do not generate characters should be put into the OnKeyDown event.

C++ Examples:

```
/*
This event handler displays a message dialog box specifying
which key was pressed:
*/

void __fastcall TForm1::FormKeyPress(TObject *Sender, char &Key)
{
  char keyString[25];
  keyString[0] = Key;
  strcpy(&keyString[1], " Was Pressed");
  TMsgDlgButtons() << mbOK;
  MessageDlg(AnsiString(keyString), TMsgDlgType(mtInformation), TMsgDlgButtons() << mbOK, 0);
}
```

Delphi Examples:

```
{
This event handler displays a message dialog box specifying
which key was pressed:
}
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  MessageDlg(Key + ' has been pressed', mtInformation, [mbOK], 0)
end;
```

See Also

OnKeyDown

OnKeyUp

KeyPreview

63.1.12.3.13 TRibbonComboBox.OnKeyUp Event

Occurs when the user releases a key that has been pressed.

Pascal

```
property OnKeyUp;
```

C++

```
__property OnKeyUp;
```

Description

Use the OnKeyUp event handler to provide special processing that occurs when a key is released. The OnKeyUp handler can

respond to all keyboard keys, including function keys and keys combined with the Shift, Alt, and Ctrl keys.

The TKeyEvent type points to a method that handles keyboard events. The Key parameter is the key on the keyboard. For non-alphanumeric keys, you must use virtual key codes to determine the key pressed. For more information, see Virtual Key codes.

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

C++ Examples:

```
/*
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
*/
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  if (Key == VK_F1)
  {
    FormColor = Form1->Color;
    Form1->Color = clAqua;
    if (ComboBox1->Text == "KeyDown")
      ComboBox1->Text = "KeyDown2";
    else
      ComboBox1->Text = "KeyDown";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}

void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key, TShiftState Shift)
{
  if (Key == VK_F1)
  {
    Form1->Color = FormColor;
    if (ComboBox1->Text == "KeyUp")
      ComboBox1->Text = "KeyUp2";
    else
      ComboBox1->Text = "KeyUp";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}
```

Delphi Examples:

```
{
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
}
var
  FormColor: TColor;
procedure TForm1.FormCreate(Sender: TObject);
begin
  KeyPreview := True;
  FormColor := Form1.Color;
```

```
end;

procedure TForm1.FormKeyDown(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := clAqua;
    if (ComboBox1.Text = 'KeyDown') then
      ComboBox1.Text := 'KeyDown2'
    else
      ComboBox1.Text := 'KeyDown';
    WriteLn(ComboBox1.Text);
  end;
end;

procedure TForm1.FormKeyUp(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := FormColor;
    if (ComboBox1.Text = 'KeyUp') then
      ComboBox1.Text := 'KeyUp2'
    else
      ComboBox1.Text := 'KeyUp';
    WriteLn(ComboBox1.Text);
  end;
end;
```

See Also

OnKeyDown

OnKeyPress

63.1.12.3.14 TRibbonComboBox.OnMouseActivate Event

Occurs when the user presses a mouse button with the mouse pointer over a control and the parent form is not active.

Pascal

```
property OnMouseActivate;
```

C++

```
__property OnMouseActivate;
```

Description

Use the OnMouseActivate event handler to implement any special processing that should occur as a result of pressing a mouse button on a control when the parent top-level form is not active.

The event type is:

```
TMouseActivateEvent = procedure (Sender: TObject; Button: TMouseButton; ShiftState:
TShiftState;
X, Y: Integer; HitTest: Integer; var MouseActivate: TMouseActivate) of object;
```

When you click on a control and the parent top-level form is not active, this event fires on the control that the mouse cursor is over. MouseActivate is a protected virtual function in TControl. MouseActivate can be overridden in custom control descendants to provide special built-in processing of the OnMouseActivate events. If you leave the default value of MouseActivate as maDefault, the parent control fires the OnMouseActivate event and repeats the process all the way up to the top-level form. If no control in the chain sets the value of MouseActivate, the behavior is the same as if a control set TMouseActivate to maActivate. At any point, a control can set the value of TMouseActivate and the parent processing stops.

Some controls set focus to themselves when they get a button down message so, in some cases, setting maNoActivate appears to have no effect. For instance, a TButton control sets focus to itself in the WM_LBUTTONDOWN message regardless of whether or not TMouseActivate is set to maNoActivate. In this case, setting maNoActivateAndEat will work because the top-level form is not activated and the WM_LBUTTONDOWN message is suppressed.

The HitTest parameter is the hit test value obtained from the WM_NCHITTEST message. See Windows.pas for valid values and their meanings. In most cases this value will be HTCLIENT, which means that the user clicked in the client area of a control. However, in the case of a top-level form, this can take other values such as HTCAPTION, or HTBOTTOM. This allows the code to decide which value to set for MouseActivate, based on the HitTest code. For example, the following code in the OnMouseActivate event handler for a TForm forces the user to click into a non-client area to activate the form:

```
if HitTest = HTCLIENT then
    MouseActivate := maNoActivateAndEat;
```

Even though this event is tied to how Windows processes WM_MOUSEACTIVATE, the actual handling of the message is done at the TControl level, which is the point in the VCL hierarchy where the control has no window handle. The OnMouseActivate event is then simulated since the all TControl descendants also have OnMouseDown, OnMouseMove, and OnMouseUp events. This information is applicable to both VCL Win32 and VCL.NET.

See Also

OnClick

OnMouseMove

OnMouseUp

MouseCapture

63.1.12.3.15 TRibbonComboBox.OnMouseDown Event

Occurs when the user presses a mouse button with the mouse pointer over a control.

Pascal

```
property OnMouseDown;
```

C++

```
__property OnMouseDown;
```

Description

Use the OnMouseDown event handler to implement any special processing that should occur as a result of pressing a mouse button.

The OnMouseDown event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. X and Y are the pixel coordinates of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
```

```
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
```

```
    StartX := X;
    StartY := Y;
  end;

  procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
  Integer);
  begin
    Form1.Canvas.Rectangle(StartX, StartY, X, Y);
    StatusBar1.Panels[0].Text := '';
    StatusBar1.Panels[1].Text := '';
    StatusBar1.Panels[2].Text := '';
    StatusBar1.Panels[3].Text := '';
  end;

  procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
  begin
    if ssLeft in Shift then { make sure button is down }
    begin
      if Y > StartY then
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
        end
      else
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
        end;
      if X > StartX then
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
        end
      else
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
        end;
    end;
  end;
```

See Also

OnClick

OnMouseMove

OnMouseUp

MouseCapture

63.1.12.3.16 TRibbonComboBox.OnMouseEnter Event

Occurs when the user moves the mouse into a control.

Pascal

```
property OnMouseEnter;
```

C++

```
__property OnMouseEnter;
```

Description

Use the OnMouseEnter event handler to implement any special processing that should occur as a result of moving the mouse into a control when the parent top-level form is not active.

63.1.12.3.17 TRibbonComboBox.OnMouseLeave Event

Occurs when the user moves the mouse outside of a control.

Pascal

```
property OnMouseLeave;
```

C++

```
__property OnMouseLeave;
```

Description

Use the OnMouseLeave event handler to implement any special processing that should occur as a result of moving the mouse outside of a control when the parent top-level form is not active.

63.1.12.3.18 TRibbonComboBox.OnMouseMove Event

Occurs when the user moves the mouse pointer while the mouse pointer is over a control.

Pascal

```
property OnMouseMove;
```

C++

```
__property OnMouseMove;
```

Description

Use the OnMouseMove event handler to respond when the mouse pointer moves after the control has captured the mouse.

Use the Shift parameter of the OnMouseDown event handler, to determine to the state of the shift keys and mouse buttons. Shift keys are the Shift, Ctrl, and Alt keys or shift key-mouse button combinations. X and Y are pixel coordinates of the new location of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
```

```
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  Form1.Canvas.Rectangle(StartX, StartY, X, Y);
  StatusBar1.Panels[0].Text := '';
  StatusBar1.Panels[1].Text := '';
  StatusBar1.Panels[2].Text := '';
  StatusBar1.Panels[3].Text := '';
end;
```

```
procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if ssLeft in Shift then { make sure button is down }
  begin
    if Y > StartY then
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
      end
    else
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
      end;
    if X > StartX then
      begin
      StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
      StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
      end
    else
      begin
      StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
      StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
      end;
  end;
end;
```

See Also

OnMouseDown

OnMouseUp

MouseCapture

63.1.12.3.19 TRibbonComboBox.OnMouseUp Event

Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.

Pascal

```
property OnMouseUp;
```

C++

```
__property OnMouseUp;
```

Description

Use an OnMouseUp event handler to implement special processing when the user releases a mouse button.

The OnMouseUp event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. X and Y are the pixel coordinates of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
```

```
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
```

```
  Y: Integer);
  begin
    StartX := X;
    StartY := Y;
  end;

  procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
  Integer);
  begin
    Form1.Canvas.Rectangle(StartX, StartY, X, Y);
    StatusBar1.Panels[0].Text := '';
    StatusBar1.Panels[1].Text := '';
    StatusBar1.Panels[2].Text := '';
    StatusBar1.Panels[3].Text := '';
  end;

  procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
  begin
    if ssLeft in Shift then { make sure button is down }
    begin
      if Y > StartY then
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
        end
      else
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
        end;
      if X > StartX then
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
        end
      else
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
        end;
    end;
  end;
```

See Also

OnMouseDown

OnMouseMove

Responding to a Mouse-up Action

63.1.12.3.20 TRibbonComboBox.OnStartDock Event

Occurs when the user begins to drag a control with a DragKind of dkDock.

Pascal

```
property OnStartDock;
```

C++

```
__property OnStartDock;
```

Description

Use the OnStartDock event handler to implement special processing when the user starts a drag-and-dock operation by dragging the control.

The OnStartDock event handler can create a TDragDockObjectEx object for the DragObject parameter to specify the appearance of the dragging rectangle and how the dragged control interacts with potential docking sites. If you return TDragDockObjectEx as the drag object, there is no need to call the Free method for the DragObject when dragging is over. If you use TDragDockObject, your application is responsible for freeing the drag object.

If the OnStartDock event handler sets the DragObject parameter to nil (Delphi) or NULL (C++), a TDragDockObject object is automatically created.

Note: This information applies to Win32 applications only.

See Also

OnEndDock

TDragDockObjectEx

DragKind

OnDragOver

OnDockDrop

63.1.12.3.21 TRibbonComboBox.OnStartDrag Event

Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

Pascal

```
property OnStartDrag;
```

C++

```
__property OnStartDrag;
```

Description

Use the OnStartDrag event handler to implement special processing when the user starts to drag the control or an object it contains. OnStartDrag only occurs if DragKind is dkDrag.

Sender is the control that is about to be dragged, or that contains the object about to be dragged.

The OnStartDrag event handler can create a TDragControlObjectEx instance for the DragObject parameter to specify the drag cursor, or, optionally, a drag image list. If you create a TDragControlObjectEx instance, there is no need to call the Free method for the DragObject when dragging is over. If you create, instead, a TDragControlObject instance, your application is responsible for freeing the drag object instance.

If the OnStartDrag event handler sets the DragObject parameter to nil (Delphi) or NULL (C++), a TDragControlObject object is automatically created and dragging begins on the control itself.

Note: On some controls, such as TRichEdit, the underlying Windows control handles internal drag operations. For these controls, there are no OnStartDrag or OnEndDrag events for drag operations within the control.

See Also

OnEndDrag

TDragControlObjectEx

DragKind

OnDragOver

OnDragDrop

63.1.13 TRibbonComboBoxItem Class

Class Hierarchy

Pascal

```
TRibbonComboBoxItem = class(TRibbonMenuItem);
```

C++

```
class TRibbonComboBoxItem : public TRibbonMenuItem;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonComboBoxItem.

63.1.13.1 TRibbonComboBoxItem Members

The following tables list the members exposed by TRibbonComboBoxItem.

Protected Methods

	Name	Description
	CalcLayout (see page 3757)	This is CalcLayout, a member of class TRibbonMenuItem.
	DrawBackground (see page 3683)	Draws a background image on the control.
	DrawGlyph (see page 3758)	Draws a symbol on the control.
	DrawSelectedBackground (see page 3758)	This is DrawSelectedBackground, a member of class TRibbonMenuItem.
	DrawSeparator (see page 3759)	This is DrawSeparator, a member of class TRibbonMenuItem.
	DrawShadowedText (see page 3759)	This is DrawShadowedText, a member of class TRibbonMenuItem.
	DrawText (see page 3683)	Draws text on the control.
	GetShowShortCut (see page 3759)	This is GetShowShortCut, a member of class TRibbonMenuItem.
	RibbonStyle (see page 3760)	This is RibbonStyle, a member of class TRibbonMenuItem.

Protected Properties

	Name	Description
	DrawBanner (see page 3760)	This is DrawBanner, a member of class TRibbonMenuItem.

Public Methods

	Name	Description
	CalcBounds (see page 3683)	Calculates the size of the menu item.
	Click (see page 3757)	Executes the control's function.
	Create (see page 3756)	Creates an instance of TCustomMenuItem.

63.1.13.2 TRibbonComboBoxItem Methods

The methods of the TRibbonComboBoxItem class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3683)	Draws a background image on the control.
	DrawText (see page 3683)	Draws text on the control.

Public Methods

	Name	Description
	CalcBounds (see page 3683)	Calculates the size of the menu item.

63.1.13.2.1 TRibbonComboBoxItem.CalcBounds Method

Calculates the size of the menu item.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as whether or not the menu item is a separator or not. CalcBounds computes the boundaries of the menu item considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.13.2.2 TRibbonComboBoxItem.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.13.2.3 TRibbonComboBoxItem.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var ARect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect ARect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

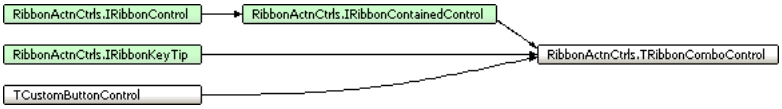
See Also

DrawBackground

DrawGlyph

63.1.14 TRibbonComboControl Class

Class Hierarchy



Pascal

```
TRibbonComboControl = class(TCustomButtonControl, IRibbonContainedControl, IRibbonKeyTip);
```

C++

```
class TRibbonComboControl : public TCustomButtonControl, public IRibbonContainedControl,
public IRibbonKeyTip;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonComboControl.

63.1.14.1 TRibbonComboControl Members

The following tables list the members exposed by TRibbonComboControl.

Protected Methods

	Name	Description
	CMBidimodechanged (see page 3688)	This is CMBidimodechanged, a member of class TRibbonComboControl.
	CMMouseenter (see page 3688)	This is CMMouseenter, a member of class TRibbonComboControl.
	CMMouseleave (see page 3688)	This is CMMouseleave, a member of class TRibbonComboControl.
	CMVisiblechanged (see page 3689)	This is CMVisiblechanged, a member of class TRibbonComboControl.
	ComboClick (see page 3689)	This is ComboClick, a member of class TRibbonComboControl.
V	DrawBackground (see page 3689)	Draws a background image on the control.
	DrawFrame (see page 3689)	This is DrawFrame, a member of class TRibbonComboControl.
V	DropDownClick (see page 3690)	This is DropDownClick, a member of class TRibbonComboControl.
	GetCaptionWidth (see page 3690)	This is GetCaptionWidth, a member of class TRibbonComboControl.
	GetGlyphWidth (see page 3690)	This is GetGlyphWidth, a member of class TRibbonComboControl.
	GetLabelWidth (see page 3691)	This is GetLabelWidth, a member of class TRibbonComboControl.
V	MouseDown (see page 3692)	OnMouseDown event dispatcher.
V	Notification (see page 3693)	Responds to notifications that components are being created or destroyed.
V	Paint (see page 3693)	Render control surface.
	RibbonStyle (see page 3694)	This is RibbonStyle, a member of class TRibbonComboControl.

	SetActionClient (see page 3694)	This is SetActionClient, a member of class TRibbonComboControl.
	SetDragMode (see page 3695)	Sets the drag mode for the control.
	SetParent (see page 3695)	Sets the parent of the control.
	ShowComboItems (see page 3696)	This is ShowComboItems, a member of class TRibbonComboControl.

Public Methods

	Name	Description
	CalcBounds (see page 3687)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3687)	This is CanAddToQuickAccessToolbar, a member of class TRibbonComboControl.
	Click (see page 3688)	Executes the control's function.
	Create (see page 3685)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3686)	Destroys an instance of TCustomActionControl.
	GetContainedControl (see page 3690)	This is GetContainedControl, a member of class TRibbonComboControl.
	GetKeyTip (see page 3690)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3691)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3692)	Hide **KeyTip** for command.
	KeyTipClick (see page 3692)	Handle clicking a **KeyTip**.
	SetBounds (see page 3694)	Sets the Left, Top, Width, and Height properties all at once.
	SetContainedControl (see page 3694)	This is SetContainedControl, a member of class TRibbonComboControl.
	SetKeyTip (see page 3695)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3696)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	ComboBox (see page 3697)	This is ComboBox, a member of class TRibbonComboControl.
	ContainedControl (see page 3697)	This is ContainedControl, a member of class TRibbonComboControl.
	KeyTip (see page 3922)	Current **KeyTip** for command
	MouseInControl (see page 3697)	This is MouseInControl, a member of class TRibbonComboControl.

63.1.14.2 TRibbonComboControl.Create Constructor

Creates an instance of TCustomButtonControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonComboControl(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomButtonControl. However, you should never attempt to instantiate a TCustomButtonControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

AOwner is the component that is responsible for freeing the button control. It becomes the value of the Owner property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Owner

Create

Destroy

63.1.14.3 TRibbonComboControl.Destroy Destructor

Destroys an instance of TCustomActionControl.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRibbonComboControl();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the graphic control reference is not nil, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

As the control is destroyed, it:

Frees the TCanvas object in its Canvas property.

Override Destroy to free any memory or resources allocated in the Create method. When declaring a Destroy method in a descendant control type, always add the override directive to the declaration and call the inherited Destroy as the last statement in the redeclared method.

When a control is created, Delphi automatically provides exception handling around the constructor call. The destructor is called if an exception escapes from the constructor. This means that the destructor code must be prepared to clean up a partially constructed instance. Check all data fields for zero before disposing of their contents.

See Also

Create

System::TObject::Free

63.1.14.4 TRibbonComboControl Methods

The methods of the TRibbonComboControl class are listed here.

Protected Methods

	Name	Description
	CMBidimodechanged (see page 3688)	This is CMBidimodechanged, a member of class TRibbonComboControl.
	CMMouseenter (see page 3688)	This is CMMouseenter, a member of class TRibbonComboControl.
	CMMouseleave (see page 3688)	This is CMMouseleave, a member of class TRibbonComboControl.
	CMVisiblechanged (see page 3689)	This is CMVisiblechanged, a member of class TRibbonComboControl.
	ComboClick (see page 3689)	This is ComboClick, a member of class TRibbonComboControl.
V	DrawBackground (see page 3689)	Draws a background image on the control.
	DrawFrame (see page 3689)	This is DrawFrame, a member of class TRibbonComboControl.
V	DropDownClick (see page 3690)	This is DropDownClick, a member of class TRibbonComboControl.
	GetCaptionWidth (see page 3690)	This is GetCaptionWidth, a member of class TRibbonComboControl.
	GetGlyphWidth (see page 3690)	This is GetGlyphWidth, a member of class TRibbonComboControl.
	GetLabelWidth (see page 3691)	This is GetLabelWidth, a member of class TRibbonComboControl.
V	MouseDown (see page 3692)	OnMouseDown event dispatcher.
V	Notification (see page 3693)	Responds to notifications that components are being created or destroyed.

	Paint (see page 3693)	Render control surface.
	RibbonStyle (see page 3694)	This is RibbonStyle, a member of class TRibbonComboControl.
	SetActionClient (see page 3694)	This is SetActionClient, a member of class TRibbonComboControl.
	SetDragMode (see page 3695)	Sets the drag mode for the control.
	SetParent (see page 3695)	Sets the parent of the control.
	ShowComboItems (see page 3696)	This is ShowComboItems, a member of class TRibbonComboControl.

Public Methods

	Name	Description
	CalcBounds (see page 3687)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3687)	This is CanAddToQuickAccessToolbar, a member of class TRibbonComboControl.
	Click (see page 3688)	Executes the control's function.
	GetContainedControl (see page 3690)	This is GetContainedControl, a member of class TRibbonComboControl.
	GetKeyTip (see page 3690)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3691)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3692)	Hide **KeyTip** for command.
	KeyTipClick (see page 3692)	Handle clicking a **KeyTip**.
	SetBounds (see page 3694)	Sets the Left, Top, Width, and Height properties all at once.
	SetContainedControl (see page 3694)	This is SetContainedControl, a member of class TRibbonComboControl.
	SetKeyTip (see page 3695)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3696)	Shows **KeyTip** for command in specified position.

63.1.14.4.1 TRibbonComboControl.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.14.4.2 TRibbonComboControl.CanAddToQuickAccessToolbar Method

Pascal

```
function CanAddToQuickAccessToolbar: Boolean;
```

C++

```
__fastcall Boolean CanAddToQuickAccessToolbar();
```

Description

This is CanAddToQuickAccessToolbar, a member of class TRibbonComboControl.

63.1.14.4.3 TRibbonComboControl.Click Method

Executes the control's function.

Pascal

```
procedure Click; override;
```

C++

```
virtual __fastcall Click();
```

Description

This override of the method inherited from TCustomActionControl is almost identical to the inherited method. The only difference is that if the button control is being manipulated at design time, extra actions are taken to ensure that the control is properly selected.

See Also

Click

Click

UsageCount

63.1.14.4.4 TRibbonComboControl.CMBidimodechanged Method

Pascal

```
procedure CMBidimodechanged(var Message: TMessage); message CM_BIDIMODECHANGED;
```

C++

```
__fastcall CMBidimodechanged(TMessage Message);
```

Description

This is CMBidimodechanged, a member of class TRibbonComboControl.

63.1.14.4.5 TRibbonComboControl.CMMouseenter Method

Pascal

```
procedure CMMouseenter(var Message: TMessage); message CM_MOUSEENTER;
```

C++

```
__fastcall CMMouseenter(TMessage Message);
```

Description

This is CMMouseenter, a member of class TRibbonComboControl.

63.1.14.4.6 TRibbonComboControl.CMMouseleave Method

Pascal

```
procedure CMMouseleave(var Message: TMessage); message CM_MOUSELEAVE;
```

C++

```
__fastcall CMMouseleave(TMessage Message);
```

Description

This is CMMouseleave, a member of class TRibbonComboControl.

63.1.14.4.7 TRibbonComboControl.CMVisiblechanged Method

Pascal

```
procedure CMVisiblechanged(var Message: TMessage); message CM_VISIBLECHANGED;
```

C++

```
__fastcall CMVisiblechanged(TMessage Message);
```

Description

This is CMVisiblechanged, a member of class TRibbonComboControl.

63.1.14.4.8 TRibbonComboControl.ComboClick Method

Pascal

```
procedure ComboClick(Sender: TObject);
```

C++

```
__fastcall ComboClick(TObject * Sender);
```

Description

This is ComboClick, a member of class TRibbonComboControl.

63.1.14.4.9 TRibbonComboControl.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.14.4.10 TRibbonComboControl.DrawFrame Method

Pascal

```
procedure DrawFrame; reintroduce;
```

C++

```
__fastcall DrawFrame();
```

Description

This is DrawFrame, a member of class TRibbonComboControl.

63.1.14.4.11 TRibbonComboControl.DropDownClick Method

Pascal

```
procedure DropDownClick(const SelectFirst: Boolean); virtual;
```

C++

```
virtual __fastcall DropDownClick(const Boolean SelectFirst);
```

Description

This is DropDownClick, a member of class TRibbonComboControl.

63.1.14.4.12 TRibbonComboControl.GetCaptionWidth Method

Pascal

```
function GetCaptionWidth: Integer;
```

C++

```
__fastcall int GetCaptionWidth();
```

Description

This is GetCaptionWidth, a member of class TRibbonComboControl.

63.1.14.4.13 TRibbonComboControl.GetContainedControl Method

Pascal

```
function GetContainedControl: TControl;
```

C++

```
__fastcall TControl * GetContainedControl();
```

Description

This is GetContainedControl, a member of class TRibbonComboControl.

63.1.14.4.14 TRibbonComboControl.GetGlyphWidth Method

Pascal

```
function GetGlyphWidth: Integer;
```

C++

```
__fastcall int GetGlyphWidth();
```

Description

This is GetGlyphWidth, a member of class TRibbonComboControl.

63.1.14.4.15 TRibbonComboControl.GetKeyTip Method

Returns current **KeyTip** for command.

Pascal

```
function GetKeyTip: string;
```

C++

```
__fastcall UnicodeString GetKeyTip();
```

Description

GetKeyTip returns the current **KeyTip** for the command

See Also

KeyTip (see page 3922)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.14.4.16 TRibbonComboControl.GetLabelWidth Method

Pascal

```
function GetLabelWidth: Integer;
```

C++

```
__fastcall int GetLabelWidth();
```

Description

This is GetLabelWidth, a member of class TRibbonComboControl.

63.1.14.4.17 TRibbonComboControl.HandleKeyTip Method

Indicates if command's **KeyTip** matches currently input KeyTip.

Pascal

```
procedure HandleKeyTip(const KeyTip: string; var Handled: TKeyTipHandled);
```

C++

```
__fastcall HandleKeyTip(const UnicodeString KeyTip, TKeyTipHandled Handled);
```

Description

HandleKeyTip indicates whether this command's

KeyTip matches the currently input KeyTip by the user. Full and partial matches are taken into consideration.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.14.4.18 TRibbonComboControl.HideKeyTip Method

Hide **KeyTip** for command.

Pascal

```
procedure HideKeyTip(const Match: string);
```

C++

```
__fastcall HideKeyTip(const UnicodeString Match);
```

Description

This method hides the **KeyTip** for this command. If the **Match** parameter matches (full or partial) the command's **KeyTip**, the **KeyTip** remains visible.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.14.4.19 TRibbonComboControl.KeyTipClick Method

Handle clicking a **KeyTip**.

Pascal

```
procedure KeyTipClick;
```

C++

```
__fastcall KeyTipClick();
```

Description

KeyTipClick Handle clicking a **KeyTip**. Some requires need to be handled differently when clicked by a **KeyTip**.

63.1.14.4.20 TRibbonComboControl.MouseDown Method

OnMouseDown event dispatcher.

Pascal

```
procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer);
override;
```

C++

```
virtual __fastcall MouseDown(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user presses the mouse button down while the cursor's hotspot is over the control.

The Button parameter determines which mouse button the user pressed. Shift indicates which shift keys (Shift, Ctrl, or Alt) were down when the user pressed the mouse button. X and Y are the pixel coordinates of the mouse pointer within the client area of

the control.

A control calls MouseDown in response to any of the Windows mouse-down messages (WM_LBUTTONDOWN, WM_MBUTTONDOWN, WM_RBUTTONDOWN), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was pressed down: left, right, or middle.

See Also

MouseMove

MouseUp

OnMouseDown

63.1.14.4.21 TRibbonComboControl.Notification Method

Responds to notifications that components are being created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification allows TControl to update its PopupMenu property when the pop-up menu it refers to is destroyed.

Override the Notification method to respond to notifications that other components are about to be destroyed or have just been created. Use the Notification method to update controls that rely on other objects. Remove references to objects that are going away so that the control does not refer to objects that have been destroyed. For example, data-aware objects override the Notification method to update themselves when their data source is removed. Some objects respond to notifications that other objects of a specific type are being created. For example, the session component sets the Session property of new data-aware controls to itself when the AutoSessionName property is true.

By default, components pass along the notification to their owned components, if any.

See Also

PopupMenu

Notification

63.1.14.4.22 TRibbonComboControl.Paint Method

Render control surface.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method is called automatically when a lightweight control needs update its display area. Code in the Paint method uses the Canvas property to render the control surface.

As implemented in TGraphicControl, Paint does nothing. Specific rendering logic is provided by Paint methods in descendant

control classes.

See Also

Canvas

63.1.14.4.23 TRibbonComboControl.RibbonStyle Method

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

This is RibbonStyle, a member of class TRibbonComboControl.

63.1.14.4.24 TRibbonComboControl.SetActionClient Method

Pascal

```
procedure SetActionClient(Value: TActionClientItem); override;
```

C++

```
virtual __fastcall SetActionClient(TActionClientItem Value);
```

Description

This is SetActionClient, a member of class TRibbonComboControl.

63.1.14.4.25 TRibbonComboControl.SetBounds Method

Sets the Left, Top, Width, and Height properties all at once.

Pascal

```
procedure SetBounds(ALeft: Integer; ATop: Integer; AWidth: Integer; AHeight: Integer);
override;
```

C++

```
virtual __fastcall SetBounds(int ALeft, int ATop, int AWidth, int AHeight);
```

Description

This override of the method inherited from TControl simultaneously sets all boundary properties at once, and thus avoiding the possibility of a control redraw between the time one value sets and another. In addition, some layout calculations are also performed.

See Also

SetBounds

63.1.14.4.26 TRibbonComboControl.SetContainedControl Method

Pascal

```
procedure SetContainedControl(const Value: TControl);
```

C++

```
__fastcall SetContainedControl(const TControl * Value);
```

Description

This is SetContainedControl, a member of class TRibbonComboControl.

63.1.14.4.27 TRibbonComboControl.SetDragMode Method

Sets the drag mode for the control.

Pascal

```
procedure SetDragMode(Value: TDragMode); override;
```

C++

```
virtual __fastcall SetDragMode(TDragMode Value);
```

Description

SetDragMode is the protected write implementation for the DragMode property. Override SetDragMode to execute additional code when the value of the DragMode property changes.

See Also

DragMode

63.1.14.4.28 TRibbonComboControl.SetKeyTip Method

Sets **KeyTip** for command.

Pascal

```
procedure SetKeyTip(const Value: string);
```

C++

```
__fastcall SetKeyTip(const UnicodeString Value);
```

Description

This method sets the **KeyTip** for the command.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

ShowKeyTip (see page 3921)

63.1.14.4.29 TRibbonComboControl.SetParent Method

Sets the parent of the control.

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

SetParent is the protected implementation of the Parent property. Override SetParent to execute additional code when the value of the Parent property changes.

The AParent parameter specifies the new parent of the control.

If the control already has a parent, SetParent removes the control from that parent's list of controls by calling the parent's RemoveControl method. If AParent is not nil (Delphi) or NULL (C++), SetParent calls its InsertControl method to add the control to its list of controls.

The TControl destructor calls SetParent(nil (Delphi) or NULL (C++)) to remove the control from its parent's control list before destroying the component.

See Also

Destroy

InsertControl

Parent

RemoveControl

63.1.14.4.30 TRibbonComboControl.ShowComboItems Method

Pascal

```
procedure ShowComboItems(const SelectFirst: Boolean); virtual;
```

C++

```
virtual __fastcall ShowComboItems(const Boolean SelectFirst);
```

Description

This is ShowComboItems, a member of class TRibbonComboControl.

63.1.14.4.31 TRibbonComboControl.ShowKeyTip Method

Shows **KeyTip** for command in specified position.

Pascal

```
procedure ShowKeyTip(const Position: TKeyTipPosition);
```

C++

```
__fastcall ShowKeyTip(const TKeyTipPosition Position);
```

Description

This method shows the **KeyTip** for the command in the position specified. Only the Application Button uses a **KeyTip** position.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

63.1.14.5 TRibbonComboControl Properties

The properties of the TRibbonComboControl class are listed here.

Public Properties

	Name	Description
	ComboBox (see page 3697)	This is ComboBox, a member of class TRibbonComboControl.
	ContainedControl (see page 3697)	This is ContainedControl, a member of class TRibbonComboControl.
	MouseInControl (see page 3697)	This is MouseInControl, a member of class TRibbonComboControl.

63.1.14.5.1 TRibbonComboControl.ComboBox Property

Pascal

```
property ComboBox: TCustomRibbonComboBox;
```

C++

```
__property TCustomRibbonComboBox ComboBox;
```

Description

This is ComboBox, a member of class TRibbonComboControl.

63.1.14.5.2 TRibbonComboControl.ContainedControl Property

Pascal

```
property ContainedControl: TControl;
```

C++

```
__property TControl * ContainedControl;
```

Description

This is ContainedControl, a member of class TRibbonComboControl.

63.1.14.5.3 TRibbonComboControl.MouseInControl Property

Pascal

```
property MouseInControl: Boolean;
```

C++

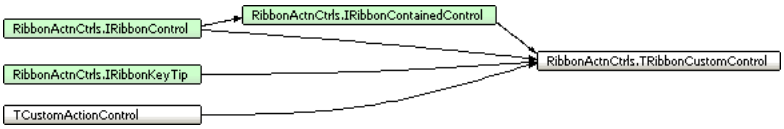
```
__property Boolean MouseInControl;
```

Description

This is MouseInControl, a member of class TRibbonComboControl.

63.1.15 TRibbonCustomControl Class

Class Hierarchy



Pascal

```
TRibbonCustomControl = class(TCustomActionControl, IRibbonControl, IRibbonContainedControl,
IRibbonKeyTip);
```

C++

```
class TRibbonCustomControl : public TCustomActionControl, public IRibbonControl, public
IRibbonContainedControl, public IRibbonKeyTip;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonCustomControl.

63.1.15.1 TRibbonCustomControl Members

The following tables list the members exposed by TRibbonCustomControl.

Protected Methods

	Name	Description
	CMVisiblechanged (see page 3701)	This is CMVisiblechanged, a member of class TRibbonCustomControl.
V	DrawBackground (see page 3702)	Draws a background image on the control.
	GetCaptionWidth (see page 3702)	This is GetCaptionWidth, a member of class TRibbonCustomControl.
	GetGlyphWidth (see page 3702)	This is GetGlyphWidth, a member of class TRibbonCustomControl.
	GetLabelWidth (see page 3703)	This is GetLabelWidth, a member of class TRibbonCustomControl.
V	Notification (see page 3704)	Responds to notifications that components are being created or destroyed.
V	Paint (see page 3705)	Render control surface.
V	SetActionClient (see page 3705)	This is SetActionClient, a member of class TRibbonCustomControl.
V	SetDragMode (see page 3708)	Sets the drag mode for the control.
V	SetParent (see page 3709)	Sets the parent of the control.

Public Methods

	Name	Description
V	CalcBounds (see page 3700)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3701)	This is CanAddToQuickAccessToolbar, a member of class TRibbonCustomControl.
V	Click (see page 3701)	Executes the control's function in response to a mouse click.
V	Create (see page 3699)	Creates an instance of TCustomActionControl.
V	Destroy (see page 3699)	Destroys an instance of TCustomActionControl.
	GetContainedControl (see page 3702)	This is GetContainedControl, a member of class TRibbonCustomControl.
	GetKeyTip (see page 3703)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3703)	Indicates if command's **KeyTip** matches currently input KeyTip.

	HideKeyTip (see page 3704)	Hide **KeyTip** for command.
	KeyTipClick (see page 3704)	Handle clicking a **KeyTip**.
	SetBounds (see page 3705)	Sets the Left, Top, Width, and Height properties all at once.
	SetContainedControl (see page 3708)	This is SetContainedControl, a member of class TRibbonCustomControl.
	SetKeyTip (see page 3708)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3709)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	ContainedControl (see page 3710)	This is ContainedControl, a member of class TRibbonCustomControl.
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.15.2 TRibbonCustomControl.Create Constructor

Creates an instance of TCustomActionControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonCustomControl(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomActionControl. However, you should never attempt to instantiate a TCustomActionControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

AOwner is the component that is responsible for freeing the TCustomActionControl instance. It becomes the value of the Owner property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Owner

Create

Destroy

63.1.15.3 TRibbonCustomControl.Destroy Destructor

Destroys an instance of TCustomActionControl.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRibbonCustomControl();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the graphic control reference is not nil, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

As the control is destroyed, it:

Frees the TCanvas object in its Canvas property.

Override Destroy to free any memory or resources allocated in the Create method. When declaring a Destroy method in a descendant control type, always add the override directive to the declaration and call the inherited Destroy as the last statement in the redeclared method.

When a control is created, Delphi automatically provides exception handling around the constructor call. The destructor is called if an exception escapes from the constructor. This means that the destructor code must be prepared to clean up a partially constructed instance. Check all data fields for zero before disposing of their contents.

See Also

Create

System::TObject::Free

63.1.15.4 TRibbonCustomControl Methods

The methods of the TRibbonCustomControl class are listed here.

Protected Methods

	Name	Description
	CMVisiblechanged (see page 3701)	This is CMVisiblechanged, a member of class TRibbonCustomControl.
V	DrawBackground (see page 3702)	Draws a background image on the control.
	GetCaptionWidth (see page 3702)	This is GetCaptionWidth, a member of class TRibbonCustomControl.
	GetGlyphWidth (see page 3702)	This is GetGlyphWidth, a member of class TRibbonCustomControl.
	GetLabelWidth (see page 3703)	This is GetLabelWidth, a member of class TRibbonCustomControl.
V	Notification (see page 3704)	Responds to notifications that components are being created or destroyed.
V	Paint (see page 3705)	Render control surface.
V	SetActionClient (see page 3705)	This is SetActionClient, a member of class TRibbonCustomControl.
V	SetDragMode (see page 3708)	Sets the drag mode for the control.
V	SetParent (see page 3709)	Sets the parent of the control.

Public Methods

	Name	Description
V	CalcBounds (see page 3700)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3701)	This is CanAddToQuickAccessToolbar, a member of class TRibbonCustomControl.
V	Click (see page 3701)	Executes the control's function in response to a mouse click.
	GetContainedControl (see page 3702)	This is GetContainedControl, a member of class TRibbonCustomControl.
	GetKeyTip (see page 3703)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3703)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3704)	Hide **KeyTip** for command.
	KeyTipClick (see page 3704)	Handle clicking a **KeyTip**.
V	SetBounds (see page 3705)	Sets the Left, Top, Width, and Height properties all at once.
	SetContainedControl (see page 3708)	This is SetContainedControl, a member of class TRibbonCustomControl.
	SetKeyTip (see page 3708)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3709)	Shows **KeyTip** for command in specified position.

63.1.15.4.1 TRibbonCustomControl.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as the presence of an icon, caption, and shortcuts. CalcBounds computes the boundaries of the control considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.15.4.2 TRibbonCustomControl.CanAddToQuickAccessToolbar Method

Pascal

```
function CanAddToQuickAccessToolbar: Boolean;
```

C++

```
__fastcall Boolean CanAddToQuickAccessToolbar();
```

Description

This is CanAddToQuickAccessToolbar, a member of class TRibbonCustomControl.

63.1.15.4.3 TRibbonCustomControl.Click Method

Executes the control's function in response to a mouse click.

Pascal

```
procedure Click; override;
```

C++

```
virtual __fastcall Click();
```

Description

This override of the method inherited from TControl executes the control's function in response to a user's mouse click. Usage data, which is used to determine whether a control is used frequently enough to be shown by default, is updated as well.

See Also

Click

UsageCount

Keyed

63.1.15.4.4 TRibbonCustomControl.CMVisiblechanged Method

Pascal

```
procedure CMVisiblechanged(var Message: TMessage); message CM_VISIBLECHANGED;
```

C++

```
__fastcall CMVisiblechanged(TMessage Message);
```

Description

This is CMVisiblechanged, a member of class TRibbonCustomControl.

63.1.15.4.5 TRibbonCustomControl.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.15.4.6 TRibbonCustomControl.GetCaptionWidth Method

Pascal

```
function GetCaptionWidth: Integer;
```

C++

```
__fastcall int GetCaptionWidth();
```

Description

This is GetCaptionWidth, a member of class TRibbonCustomControl.

63.1.15.4.7 TRibbonCustomControl.GetContainedControl Method

Pascal

```
function GetContainedControl: TControl;
```

C++

```
__fastcall TControl * GetContainedControl();
```

Description

This is GetContainedControl, a member of class TRibbonCustomControl.

63.1.15.4.8 TRibbonCustomControl.GetGlyphWidth Method

Pascal

```
function GetGlyphWidth: Integer;
```

C++

```
__fastcall int GetGlyphWidth();
```

Description

This is GetGlyphWidth, a member of class TRibbonCustomControl.

63.1.15.4.9 TRibbonCustomControl.GetKeyTip Method

Returns current **KeyTip** for command.

Pascal

```
function GetKeyTip: string;
```

C++

```
__fastcall UnicodeString GetKeyTip();
```

Description

GetKeyTip returns the current **KeyTip** for the command

See Also

KeyTip (see page 3922)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.15.4.10 TRibbonCustomControl.GetLabelWidth Method

Pascal

```
function GetLabelWidth: Integer;
```

C++

```
__fastcall int GetLabelWidth();
```

Description

This is GetLabelWidth, a member of class TRibbonCustomControl.

63.1.15.4.11 TRibbonCustomControl.HandleKeyTip Method

Indicates if command's **KeyTip** matches currently input KeyTip.

Pascal

```
procedure HandleKeyTip(const KeyTip: string; var Handled: TKeyTipHandled);
```

C++

```
__fastcall HandleKeyTip(const UnicodeString KeyTip, TKeyTipHandled Handled);
```

Description

HandleKeyTip indicates whether this command's

KeyTip matches the currently input KeyTip by the user. Full and partial matches are taken into consideration.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.15.4.12 TRibbonCustomControl.HideKeyTip Method

Hide **KeyTip** for command.

Pascal

```
procedure HideKeyTip(const Match: string);
```

C++

```
__fastcall HideKeyTip(const UnicodeString Match);
```

Description

This method hides the **KeyTip** for this command. If the **Match** parameter matches (full or partial) the command's **KeyTip**, the **KeyTip** remains visible.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.1.15.4.13 TRibbonCustomControl.KeyTipClick Method

Handle clicking a **KeyTip**.

Pascal

```
procedure KeyTipClick;
```

C++

```
__fastcall KeyTipClick();
```

Description

KeyTipClick Handle clicking a **KeyTip**. Some requires need to be handled differently when clicked by a **KeyTip**.

63.1.15.4.14 TRibbonCustomControl.Notification Method

Responds to notifications that components are being created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification allows TControl to update its PopupMenu property when the pop-up menu it refers to is destroyed.

Override the Notification method to respond to notifications that other components are about to be destroyed or have just been created. Use the Notification method to update controls that rely on other objects. Remove references to objects that are going away so that the control does not refer to objects that have been destroyed. For example, data-aware objects override the Notification method to update themselves when their data source is removed. Some objects respond to notifications that other objects of a specific type are being created. For example, the session component sets the Session property of new data-aware controls to itself when the AutoSessionName property is true.

By default, components pass along the notification to their owned components, if any.

See Also

PopupMenu

Notification

63.1.15.4.15 TRibbonCustomControl.Paint Method

Render control surface.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method is called automatically when a lightweight control needs update its display area. Code in the Paint method uses the Canvas property to render the control surface.

As implemented in TGraphicControl, Paint does nothing. Specific rendering logic is provided by Paint methods in descendant control classes.

See Also

Canvas

63.1.15.4.16 TRibbonCustomControl.SetActionClient Method

Pascal

```
procedure SetActionClient(Value: TActionClientItem); override;
```

C++

```
virtual __fastcall SetActionClient(TActionClientItem Value);
```

Description

This is SetActionClient, a member of class TRibbonCustomControl.

63.1.15.4.17 TRibbonCustomControl.SetBounds Method

Sets the Left, Top, Width, and Height properties all at once.

Pascal

```
procedure SetBounds(ALeft: Integer; ATop: Integer; AWidth: Integer; AHeight: Integer);
override;
```

C++

```
virtual __fastcall SetBounds(int ALeft, int ATop, int AWidth, int AHeight);
```

Description

Use SetBounds to change all of the component's boundary properties at one time. The same effect can be achieved by setting the Left, Top, Width, and Height properties separately, but SetBounds changes all four properties at once ensuring that the control will not repaint between changes.

Specify the values for the Left, Top, Width, and Height properties as the value of the ALeft, ATop, AWidth, and AHeight parameters, respectively.

Calling SetBounds does not necessarily result in the Left, Top, Width, and Height properties changing to the specified values. Before the properties are changed, the AutoSize or Constraints property may limit the changes, and an OnCanResize (or OnConstrainedResize) event handler may change the new values. After the control's Left, Top, Width, and Height properties are changed, SetBounds generates an OnResize event.

Note: Component writers can change the Left, Top, Width, and Height properties while bypassing all resize events and constraint or autosize logic by using the UpdateBoundsRect method instead.

C++ Examples:

```
//
//This example demostrates the usage of Margins and AlignWithMargins
//properties. Note that Margins will only work when AlignWithMargins
//is set to true and Align is not alNone.
//

#define TopPanelMargin 10
#define BottomPanelMargin 20

TPanel *TopPanel, *BottomPanel;

void __fastcall TForm2::FormCreate(TObject *Sender)
{
    /* Create a new panel and align it to top */
    TopPanel = new TPanel(this);
    TopPanel->Parent = this;
    TopPanel->Align = alTop;
    TopPanel->Height = (ClientHeight / 2);

    /* Create a new panel and align it to the remaining client size */
    BottomPanel = new TPanel(this);
    BottomPanel->Parent = this;
    BottomPanel->Align = alClient;

    /* Set Top panel margins */
    /* Note: Margins will not have an effect if Align is alNone */
    TopPanel->Margins->SetBounds(TopPanelMargin, TopPanelMargin,
                                 TopPanelMargin, TopPanelMargin);
    TopPanel->AlignWithMargins = true;
    TopPanel->Caption = "TopPanel";

    /* Set Top Bottom margins */
    /* Note: Margins will not have an effect if Align is alNone */
    BottomPanel->Margins->SetBounds(BottomPanelMargin, BottomPanelMargin,
                                    BottomPanelMargin, BottomPanelMargin);
    BottomPanel->AlignWithMargins = true;
```

```
    BottomPanel->Caption = "BottomPanel";

    if ((TopPanel->Top != TopPanelMargin) ||
       (TopPanel->Left != TopPanelMargin) ||
       (TopPanel->Width != (ClientWidth - (TopPanelMargin * 2))) ||
       (BottomPanel->Top != (TopPanel->Height + (TopPanelMargin * 2) +
                                                BottomPanelMargin)) ||
       (BottomPanel->Left != BottomPanelMargin) ||
       (BottomPanel->Width != (ClientWidth - (BottomPanelMargin * 2))))
       {
          MessageDlg(AnsiString(
            "This should not happen! Position and size are ")+
            "calculated relative to the margins",
            mtError, TMsgDlgButtons() << mbOK, 0);
       }
}
```

Delphi Examples:

```
{
This example demostrates the usage of Margins and AlignWithMargins
properties. Note that Margins will only work when AlignWithMargins
is set to true and Align is not alNone.
}
const
  TopPanelMargin = 10;
  BottomPanelMargin = 30;

var
  TopPanel, BottomPanel : TPanel;

procedure TForm2.FormCreate(Sender: TObject);
begin
  { Create a new panel and align it to top }
  TopPanel := TPanel.Create(Self);
  TopPanel.Parent := Self;
  TopPanel.Align := alTop;
  TopPanel.Height := (ClientHeight div 2);

  { Create a new panel and align it to the remaining client size }
  BottomPanel := TPanel.Create(Self);
  BottomPanel.Parent := Self;
  BottomPanel.Align := alClient;

  { Set Top panel margins }
  { Note: Margins will not have an effect if Align is alNone }
  TopPanel.Margins.SetBounds(TopPanelMargin, TopPanelMargin,
                             TopPanelMargin, TopPanelMargin);
  TopPanel.AlignWithMargins := true;
  TopPanel.Caption:= 'TopPanel';

  { Set Bottom panel margins }
  { Note: Margins will not have an effect if Align is alNone }
  BottomPanel.Margins.SetBounds(BottomPanelMargin, BottomPanelMargin,
                                BottomPanelMargin, BottomPanelMargin);
  BottomPanel.AlignWithMargins := true;
  BottomPanel.Caption:= 'BottomPanel';

  if (TopPanel.Top <> TopPanelMargin) or
     (TopPanel.Left <> TopPanelMargin) or
     (TopPanel.Width <> (ClientWidth - (TopPanelMargin * 2))) or
     (BottomPanel.Top <> (TopPanel.Height + (TopPanelMargin * 2) +
                                              BottomPanelMargin)) or
     (BottomPanel.Left <> BottomPanelMargin) or
     (BottomPanel.Width <> (ClientWidth - (BottomPanelMargin * 2)))
     then
       MessageDlg('This should not happen! Position and size are ' +
```

```
            'calculated relative to the margins', mtError, [mbOK], 0);
  end;
```

See Also

Height

Left

Top

Width

BoundsRect

AutoSize

Constraints

OnCanResize

OnConstrainedResize

OnResize

UpdateBoundsRect

63.1.15.4.18 TRibbonCustomControl.SetContainedControl Method

Pascal

```
procedure SetContainedControl(const Value: TControl);
```

C++

```
__fastcall SetContainedControl(const TControl * Value);
```

Description

This is SetContainedControl, a member of class TRibbonCustomControl.

63.1.15.4.19 TRibbonCustomControl.SetDragMode Method

Sets the drag mode for the control.

Pascal

```
procedure SetDragMode(Value: TDragMode); override;
```

C++

```
virtual __fastcall SetDragMode(TDragMode Value);
```

Description

SetDragMode is the protected write implementation for the DragMode property. Override SetDragMode to execute additional code when the value of the DragMode property changes.

See Also

DragMode

63.1.15.4.20 TRibbonCustomControl.SetKeyTip Method

Sets **KeyTip** for command.

Pascal

```
procedure SetKeyTip(const Value: string);
```

C++

```
__fastcall SetKeyTip(const UnicodeString Value);
```

Description

This method sets the **KeyTip** for the command.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

ShowKeyTip (see page 3921)

63.1.15.4.21 TRibbonCustomControl.SetParent Method

Sets the parent of the control.

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

SetParent is the protected implementation of the Parent property. Override SetParent to execute additional code when the value of the Parent property changes.

The AParent parameter specifies the new parent of the control.

If the control already has a parent, SetParent removes the control from that parent's list of controls by calling the parent's RemoveControl method. If AParent is not nil (Delphi) or NULL (C++), SetParent calls its InsertControl method to add the control to its list of controls.

The TControl destructor calls SetParent(nil (Delphi) or NULL (C++)) to remove the control from its parent's control list before destroying the component.

See Also

Destroy

InsertControl

Parent

RemoveControl

63.1.15.4.22 TRibbonCustomControl.ShowKeyTip Method

Shows **KeyTip** for command in specified position.

Pascal

```
procedure ShowKeyTip(const Position: TKeyTipPosition);
```

C++

```
__fastcall ShowKeyTip(const TKeyTipPosition Position);
```

Description

This method shows the **KeyTip** for the command in the position specified. Only the Application Button uses a **KeyTip** position.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

63.1.15.5 TRibbonCustomControl Properties

The properties of the TRibbonCustomControl class are listed here.

Public Properties

	Name	Description
	ContainedControl (see page 3710)	This is ContainedControl, a member of class TRibbonCustomControl.

63.1.15.5.1 TRibbonCustomControl.ContainedControl Property

Pascal

```
property ContainedControl: TControl;
```

C++

```
__property TControl * ContainedControl;
```

Description

This is ContainedControl, a member of class TRibbonCustomControl.

63.1.16 TRibbonCustomizeItem Class

Class Hierarchy

Pascal

```
TRibbonCustomizeItem = class(TCustomAddRemoveItem);
```

C++

```
class TRibbonCustomizeItem : public TCustomAddRemoveItem;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonCustomizeItem.

63.1.16.1 TRibbonCustomizeItem Members

The following tables list the members exposed by TRibbonCustomizeItem.

Protected Methods

	Name	Description
	DrawBackground (see page 3712)	Draws a background image on the control.
	DrawGlyph (see page 3712)	Draws a symbol on the control.
	RibbonStyle (see page 3712)	This is RibbonStyle, a member of class TRibbonCustomizeItem.

Public Methods

	Name	Description
	CalcBounds (see page 3711)	Calculates the size of the add/remove menu item.

63.1.16.2 TRibbonCustomizeItem Methods

The methods of the TRibbonCustomizeItem class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3712)	Draws a background image on the control.
	DrawGlyph (see page 3712)	Draws a symbol on the control.
	RibbonStyle (see page 3712)	This is RibbonStyle, a member of class TRibbonCustomizeItem.

Public Methods

	Name	Description
	CalcBounds (see page 3711)	Calculates the size of the add/remove menu item.

63.1.16.2.1 TRibbonCustomizeItem.CalcBounds Method

Calculates the size of the add/remove menu item.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

CalcBounds updates the Height, Width, and TextBounds properties to reflect current conditions (such as the item's caption and the current font).

See Also

Height

Width

TextBounds

63.1.16.2.2 TRibbonCustomizeItem.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.16.2.3 TRibbonCustomizeItem.DrawGlyph Method

Draws a symbol on the control.

Pascal

```
procedure DrawGlyph(const Location: TPoint); override;
```

C++

```
virtual __fastcall DrawGlyph(const TPoint Location);
```

Description

Use DrawGlyph to draw a symbol on the control at the specified location.

See Also

DrawBackground

DrawText

63.1.16.2.4 TRibbonCustomizeItem.RibbonStyle Method

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

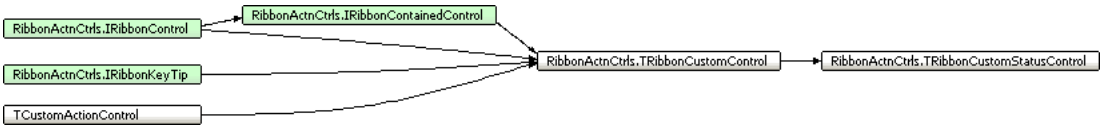
```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

This is RibbonStyle, a member of class TRibbonCustomizeItem.

63.1.17 TRibbonCustomStatusControl Class

Class Hierarchy



Pascal

```
TRibbonCustomStatusControl = class(TRibbonCustomControl);
```

C++

```
class TRibbonCustomStatusControl : public TRibbonCustomControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonCustomStatusControl.

63.1.17.1 TRibbonCustomStatusControl Members

The following tables list the members exposed by TRibbonCustomStatusControl.

Protected Methods

	Name	Description
	CMVisiblechanged (see page 3701)	This is CMVisiblechanged, a member of class TRibbonCustomControl.
	DrawBackground (see page 3715)	Draws a background image on the control.
	GetCaptionWidth (see page 3702)	This is GetCaptionWidth, a member of class TRibbonCustomControl.
	GetGlyphWidth (see page 3702)	This is GetGlyphWidth, a member of class TRibbonCustomControl.
	GetLabelWidth (see page 3703)	This is GetLabelWidth, a member of class TRibbonCustomControl.
	Notification (see page 3704)	Responds to notifications that components are being created or destroyed.
	Paint (see page 3705)	Render control surface.
	SetActionClient (see page 3705)	This is SetActionClient, a member of class TRibbonCustomControl.
	SetDragMode (see page 3708)	Sets the drag mode for the control.
	SetParent (see page 3709)	Sets the parent of the control.

Public Methods

	Name	Description
	CalcBounds (see page 3714)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3701)	This is CanAddToQuickAccessToolbar, a member of class TRibbonCustomControl.
	Click (see page 3701)	Executes the control's function in response to a mouse click.
	Create (see page 3699)	Creates an instance of TCustomActionControl.
	Destroy (see page 3699)	Destroys an instance of TCustomActionControl.
	GetContainedControl (see page 3915)	This is GetContainedControl, a member of class IRibbonContainedControl.
	GetContainedControl (see page 3702)	This is GetContainedControl, a member of class TRibbonCustomControl.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3703)	Returns current **KeyTip** for command.

	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3703)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3704)	Hide **KeyTip** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
	KeyTipClick (see page 3704)	Handle clicking a **KeyTip**.
	SetBounds (see page 3705)	Sets the Left, Top, Width, and Height properties all at once.
	SetContainedControl (see page 3915)	This is SetContainedControl, a member of class IRibbonContainedControl.
	SetContainedControl (see page 3708)	This is SetContainedControl, a member of class TRibbonCustomControl.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3708)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3709)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	ContainedControl (see page 3915)	This is ContainedControl, a member of class IRibbonContainedControl.
	ContainedControl (see page 3710)	This is ContainedControl, a member of class TRibbonCustomControl.
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.17.2 TRibbonCustomStatusControl Methods

The methods of the TRibbonCustomStatusControl class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3715)	Draws a background image on the control.

Public Methods

	Name	Description
	CalcBounds (see page 3714)	Calculates the size of the control.

63.1.17.2.1 TRibbonCustomStatusControl.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as the presence of an icon, caption, and shortcuts. CalcBounds computes the boundaries of the control considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.17.2.2 TRibbonCustomStatusControl.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.18 TRibbonDropDownButton Class

Class Hierarchy

Pascal

```
TRibbonDropDownButton = class(TRibbonBaseButtonControl);
```

C++

```
class TRibbonDropDownButton : public TRibbonBaseButtonControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonDropDownButton.

63.1.18.1 TRibbonDropDownButton Members

The following tables list the members exposed by TRibbonDropDownButton.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	CalcLayout (see page 3717)	This is CalcLayout, a member of class TRibbonDropDownButton.
	DoDropDown (see page 3718)	This is DoDropDown, a member of class TRibbonDropDownButton.
	DrawArrow (see page 3718)	This is DrawArrow, a member of class TRibbonDropDownButton.
	DrawBackground (see page 3718)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.

	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3718)	This is DrawShadowedText, a member of class TRibbonDropDownButton.
	DrawText (see page 3719)	Draws text on the control.
	GetArrowPosition (see page 3719)	This is GetArrowPosition, a member of class TRibbonDropDownButton.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	GetPopupClass (see page 3719)	This is GetPopupClass, a member of class TRibbonDropDownButton.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3717)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3717)	Executes the control's function.
	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3719)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.18.2 TRibbonDropDownButton Methods

The methods of the TRibbonDropDownButton class are listed here.

Protected Methods

	Name	Description
	CalcLayout (see page 3717)	This is CalcLayout, a member of class TRibbonDropDownButton.
	DoDropDown (see page 3718)	This is DoDropDown, a member of class TRibbonDropDownButton.
	DrawArrow (see page 3718)	This is DrawArrow, a member of class TRibbonDropDownButton.
	DrawBackground (see page 3718)	Draws a background image on the control.
	DrawShadowedText (see page 3718)	This is DrawShadowedText, a member of class TRibbonDropDownButton.
	DrawText (see page 3719)	Draws text on the control.
	GetArrowPosition (see page 3719)	This is GetArrowPosition, a member of class TRibbonDropDownButton.

	GetPopupClass (see page 3719)	This is GetPopupClass, a member of class TRibbonDropDownButton.

Public Methods

	Name	Description
	CalcBounds (see page 3717)	Calculates the size of the control.
	Click (see page 3717)	Executes the control's function.
	KeyTipClick (see page 3719)	Handle clicking a **KeyTip**.

63.1.18.2.1 TRibbonDropDownButton.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.18.2.2 TRibbonDropDownButton.CalcLayout Method

Pascal

```
procedure CalcLayout; override;
```

C++

```
virtual __fastcall CalcLayout();
```

Description

This is CalcLayout, a member of class TRibbonDropDownButton.

63.1.18.2.3 TRibbonDropDownButton.Click Method

Executes the control's function.

Pascal

```
procedure Click; override;
```

C++

```
virtual __fastcall Click();
```

Description

This override of the method inherited from TCustomActionControl is almost identical to the inherited method. The only difference is that if the button control is being manipulated at design time, extra actions are taken to ensure that the control is properly selected.

See Also

Click

Click

UsageCount

63.1.18.2.4 TRibbonDropDownButton.DoDropDown Method

Pascal

```
procedure DoDropDown(const SelectFirst: Boolean); virtual;
```

C++

```
virtual __fastcall DoDropDown(const Boolean SelectFirst);
```

Description

This is DoDropDown, a member of class TRibbonDropDownButton.

63.1.18.2.5 TRibbonDropDownButton.DrawArrow Method

Pascal

```
procedure DrawArrow(Location: TPoint; Color: TColor; ShadowColor: TColor; Size: Integer);
virtual;
```

C++

```
virtual __fastcall DrawArrow(TPoint Location, TColor Color, TColor ShadowColor, int Size);
```

Description

This is DrawArrow, a member of class TRibbonDropDownButton.

63.1.18.2.6 TRibbonDropDownButton.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.18.2.7 TRibbonDropDownButton.DrawShadowedText Method

Pascal

```
procedure DrawShadowedText(Rect: TRect; Flags: Cardinal; Text: string; TextColor: TColor;
ShadowColor: TColor); override;
```

C++

```
virtual __fastcall DrawShadowedText(TRect Rect, unsigned Flags, UnicodeString Text, TColor
TextColor, TColor ShadowColor);
```

Description

This is DrawShadowedText, a member of class TRibbonDropDownButton.

63.1.18.2.8 TRibbonDropDownButton.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var ARect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect ARect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

See Also

DrawBackground

DrawGlyph

63.1.18.2.9 TRibbonDropDownButton.GetArrowPosition Method

Pascal

```
function GetArrowPosition(const ARect: TRect): TPoint; virtual;
```

C++

```
virtual __fastcall TPoint GetArrowPosition(const TRect ARect);
```

Description

This is GetArrowPosition, a member of class TRibbonDropDownButton.

63.1.18.2.10 TRibbonDropDownButton.GetPopupClass Method

Pascal

```
function GetPopupClass: TCustomPopupClass; virtual;
```

C++

```
virtual __fastcall TCustomPopupClass GetPopupClass();
```

Description

This is GetPopupClass, a member of class TRibbonDropDownButton.

63.1.18.2.11 TRibbonDropDownButton.KeyTipClick Method

Handle clicking a **KeyTip**.

Pascal

```
procedure KeyTipClick; override;
```

C++

```
virtual __fastcall KeyTipClick();
```

Description

KeyTipClick Handle clicking a **KeyTip**. Some requires need to be handled differently when clicked by a **KeyTip**.

63.1.19 TRibbonGallery Class

Class Hierarchy

Pascal

```
TRibbonGallery = class(TControl);
```

C++

```
class TRibbonGallery : public TControl;
```

File

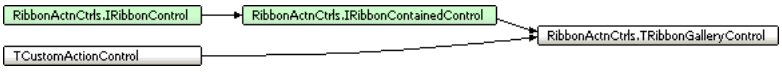
RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonGallery.

63.1.20 TRibbonGalleryControl Class

Class Hierarchy



Pascal

```
TRibbonGalleryControl = class(TCustomActionControl, IRibbonContainedControl);
```

C++

```
class TRibbonGalleryControl : public TCustomActionControl, public IRibbonContainedControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonGalleryControl.

63.1.20.1 TRibbonGalleryControl Members

The following tables list the members exposed by TRibbonGalleryControl.

Public Methods

	Name	Description
	CanAddToQuickAccessToolbar (see page 3721)	This is CanAddToQuickAccessToolbar, a member of class TRibbonGalleryControl.

	GetContainedControl (see page 3721)	This is GetContainedControl, a member of class TRibbonGalleryControl.
	SetContainedControl (see page 3721)	This is SetContainedControl, a member of class TRibbonGalleryControl.

Public Properties

	Name	Description
	ContainedControl (see page 3722)	This is ContainedControl, a member of class TRibbonGalleryControl.
R	Gallery (see page 3722)	This is Gallery, a member of class TRibbonGalleryControl.

63.1.20.2 TRibbonGalleryControl Methods

The methods of the TRibbonGalleryControl class are listed here.

Public Methods

	Name	Description
	CanAddToQuickAccessToolbar (see page 3721)	This is CanAddToQuickAccessToolbar, a member of class TRibbonGalleryControl.
	GetContainedControl (see page 3721)	This is GetContainedControl, a member of class TRibbonGalleryControl.
	SetContainedControl (see page 3721)	This is SetContainedControl, a member of class TRibbonGalleryControl.

63.1.20.2.1 TRibbonGalleryControl.CanAddToQuickAccessToolbar Method

Pascal

```
function CanAddToQuickAccessToolbar: Boolean;
```

C++

```
__fastcall Boolean CanAddToQuickAccessToolbar();
```

Description

This is CanAddToQuickAccessToolbar, a member of class TRibbonGalleryControl.

63.1.20.2.2 TRibbonGalleryControl.GetContainedControl Method

Pascal

```
function GetContainedControl: TControl;
```

C++

```
__fastcall TControl * GetContainedControl();
```

Description

This is GetContainedControl, a member of class TRibbonGalleryControl.

63.1.20.2.3 TRibbonGalleryControl.SetContainedControl Method

Pascal

```
procedure SetContainedControl(const Value: TControl);
```

C++

```
__fastcall SetContainedControl(const TControl * Value);
```

Description

This is SetContainedControl, a member of class TRibbonGalleryControl.

63.1.20.3 TRibbonGalleryControl Properties

The properties of the TRibbonGalleryControl class are listed here.

Public Properties

	Name	Description
	ContainedControl (see page 3722)	This is ContainedControl, a member of class TRibbonGalleryControl.
R	Gallery (see page 3722)	This is Gallery, a member of class TRibbonGalleryControl.

63.1.20.3.1 TRibbonGalleryControl.ContainedControl Property

Pascal

```
property ContainedControl: TControl;
```

C++

```
__property TControl * ContainedControl;
```

Description

This is ContainedControl, a member of class TRibbonGalleryControl.

63.1.20.3.2 TRibbonGalleryControl.Gallery Property

Pascal

```
property Gallery: TRibbonGallery;
```

C++

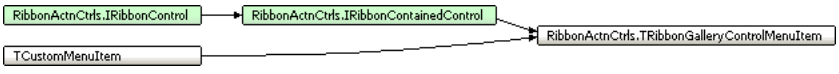
```
__property TRibbonGallery Gallery;
```

Description

This is Gallery, a member of class TRibbonGalleryControl.

63.1.21 TRibbonGalleryControlMenuItem Class

Class Hierarchy



Pascal

```
TRibbonGalleryControlMenuItem = class(TCustomMenuItem, IRibbonContainedControl);
```

C++

```
class TRibbonGalleryControlMenuItem : public TCustomMenuItem, public IRibbonContainedControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonGalleryControlMenuItem.

63.1.21.1 TRibbonGalleryControlMenuItem Members

The following tables list the members exposed by TRibbonGalleryControlMenuItem.

Protected Methods

	Name	Description
	Paint (see page 3724)	Render control surface.
	SetParent (see page 3727)	Sets the parent of the control.

Public Methods

	Name	Description
	CalcBounds (see page 3723)	Calculates the size of the menu item.
	CanAddToQuickAccessToolbar (see page 3724)	This is CanAddToQuickAccessToolbar, a member of class TRibbonGalleryControlMenuItem.
	GetContainedControl (see page 3724)	This is GetContainedControl, a member of class TRibbonGalleryControlMenuItem.
	SetBounds (see page 3725)	Sets the Left, Top, Width, and Height properties all at once.
	SetContainedControl (see page 3727)	This is SetContainedControl, a member of class TRibbonGalleryControlMenuItem.

Public Properties

	Name	Description
	ContainedControl (see page 3728)	This is ContainedControl, a member of class TRibbonGalleryControlMenuItem.

63.1.21.2 TRibbonGalleryControlMenuItem Methods

The methods of the TRibbonGalleryControlMenuItem class are listed here.

Protected Methods

	Name	Description
	Paint (see page 3724)	Render control surface.
	SetParent (see page 3727)	Sets the parent of the control.

Public Methods

	Name	Description
	CalcBounds (see page 3723)	Calculates the size of the menu item.
	CanAddToQuickAccessToolbar (see page 3724)	This is CanAddToQuickAccessToolbar, a member of class TRibbonGalleryControlMenuItem.
	GetContainedControl (see page 3724)	This is GetContainedControl, a member of class TRibbonGalleryControlMenuItem.
	SetBounds (see page 3725)	Sets the Left, Top, Width, and Height properties all at once.
	SetContainedControl (see page 3727)	This is SetContainedControl, a member of class TRibbonGalleryControlMenuItem.

63.1.21.2.1 TRibbonGalleryControlMenuItem.CalcBounds Method

Calculates the size of the menu item.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as whether or not the menu item is a separator or not. CalcBounds computes the boundaries of the menu item considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.21.2.2 TRibbonGalleryControlMenuItem.CanAddToQuickAccessToolbar Method

Pascal

```
function CanAddToQuickAccessToolbar: Boolean;
```

C++

```
__fastcall Boolean CanAddToQuickAccessToolbar();
```

Description

This is CanAddToQuickAccessToolbar, a member of class TRibbonGalleryControlMenuItem.

63.1.21.2.3 TRibbonGalleryControlMenuItem.GetContainedControl Method

Pascal

```
function GetContainedControl: TControl;
```

C++

```
__fastcall TControl * GetContainedControl();
```

Description

This is GetContainedControl, a member of class TRibbonGalleryControlMenuItem.

63.1.21.2.4 TRibbonGalleryControlMenuItem.Paint Method

Render control surface.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method is called automatically when a lightweight control needs update its display area. Code in the Paint method uses the Canvas property to render the control surface.

As implemented in TGraphicControl, Paint does nothing. Specific rendering logic is provided by Paint methods in descendant control classes.

See Also

Canvas

63.1.21.2.5 TRibbonGalleryControlMenuItem.SetBounds Method

Sets the Left, Top, Width, and Height properties all at once.

Pascal

```
procedure SetBounds(ALeft: Integer; ATop: Integer; AWidth: Integer; AHeight: Integer);
override;
```

C++

```
virtual __fastcall SetBounds(int ALeft, int ATop, int AWidth, int AHeight);
```

Description

Use SetBounds to change all of the component's boundary properties at one time. The same effect can be achieved by setting the Left, Top, Width, and Height properties separately, but SetBounds changes all four properties at once ensuring that the control will not repaint between changes.

Specify the values for the Left, Top, Width, and Height properties as the value of the ALeft, ATop, AWidth, and AHeight parameters, respectively.

Calling SetBounds does not necessarily result in the Left, Top, Width, and Height properties changing to the specified values. Before the properties are changed, the AutoSize or Constraints property may limit the changes, and an OnCanResize (or OnConstrainedResize) event handler may change the new values. After the control's Left, Top, Width, and Height properties are changed, SetBounds generates an OnResize event.

Note: Component writers can change the Left, Top, Width, and Height properties while bypassing all resize events and constraint or autosize logic by using the UpdateBoundsRect method instead.

C++ Examples:

```
//
//This example demostrates the usage of Margins and AlignWithMargins
//properties. Note that Margins will only work when AlignWithMargins
//is set to true and Align is not alNone.
//

#define TopPanelMargin 10
#define BottomPanelMargin 20

TPanel *TopPanel, *BottomPanel;

void __fastcall TForm2::FormCreate(TObject *Sender)
{
    /* Create a new panel and align it to top */
    TopPanel = new TPanel(this);
    TopPanel->Parent = this;
    TopPanel->Align = alTop;
    TopPanel->Height = (ClientHeight / 2);

    /* Create a new panel and align it to the remaining client size */
    BottomPanel = new TPanel(this);
    BottomPanel->Parent = this;
    BottomPanel->Align = alClient;

    /* Set Top panel margins */
    /* Note: Margins will not have an effect if Align is alNone */
    TopPanel->Margins->SetBounds(TopPanelMargin, TopPanelMargin,
                                 TopPanelMargin, TopPanelMargin);
    TopPanel->AlignWithMargins = true;
    TopPanel->Caption = "TopPanel";
```

```
    /* Set Top Bottom margins */
    /* Note: Margins will not have an effect if Align is alNone */
    BottomPanel->Margins->SetBounds(BottomPanelMargin, BottomPanelMargin,
                                    BottomPanelMargin, BottomPanelMargin);
    BottomPanel->AlignWithMargins = true;
    BottomPanel->Caption = "BottomPanel";

    if ((TopPanel->Top != TopPanelMargin) ||
       (TopPanel->Left != TopPanelMargin) ||
       (TopPanel->Width != (ClientWidth - (TopPanelMargin * 2))) ||
       (BottomPanel->Top != (TopPanel->Height + (TopPanelMargin * 2) +
                                               BottomPanelMargin)) ||
       (BottomPanel->Left != BottomPanelMargin) ||
       (BottomPanel->Width != (ClientWidth - (BottomPanelMargin * 2))))
       {
          MessageDlg(AnsiString(
            "This should not happen! Position and size are ")+
            "calculated relative to the margins",
            mtError, TMsgDlgButtons() << mbOK, 0);
       }
}
```

Delphi Examples:

```
{
This example demostrates the usage of Margins and AlignWithMargins
properties. Note that Margins will only work when AlignWithMargins
is set to true and Align is not alNone.
}
const
  TopPanelMargin = 10;
  BottomPanelMargin = 30;

var
  TopPanel, BottomPanel : TPanel;

procedure TForm2.FormCreate(Sender: TObject);
begin
  { Create a new panel and align it to top }
  TopPanel := TPanel.Create(Self);
  TopPanel.Parent := Self;
  TopPanel.Align := alTop;
  TopPanel.Height := (ClientHeight div 2);

  { Create a new panel and align it to the remaining client size }
  BottomPanel := TPanel.Create(Self);
  BottomPanel.Parent := Self;
  BottomPanel.Align := alClient;

  { Set Top panel margins }
  { Note: Margins will not have an effect if Align is alNone }
  TopPanel.Margins.SetBounds(TopPanelMargin, TopPanelMargin,
                             TopPanelMargin, TopPanelMargin);
  TopPanel.AlignWithMargins := true;
  TopPanel.Caption:= 'TopPanel';

  { Set Bottom panel margins }
  { Note: Margins will not have an effect if Align is alNone }
  BottomPanel.Margins.SetBounds(BottomPanelMargin, BottomPanelMargin,
                                BottomPanelMargin, BottomPanelMargin);
  BottomPanel.AlignWithMargins := true;
  BottomPanel.Caption:= 'BottomPanel';

  if (TopPanel.Top <> TopPanelMargin) or
     (TopPanel.Left <> TopPanelMargin) or
     (TopPanel.Width <> (ClientWidth - (TopPanelMargin * 2))) or
```

```
      (BottomPanel.Top <> (TopPanel.Height + (TopPanelMargin * 2) +
                                              BottomPanelMargin)) or
      (BottomPanel.Left <> BottomPanelMargin) or
      (BottomPanel.Width <> (ClientWidth - (BottomPanelMargin * 2)))
      then
        MessageDlg('This should not happen! Position and size are ' +
             'calculated relative to the margins', mtError, [mbOK], 0);
end;
```

See Also

Height

Left

Top

Width

BoundsRect

AutoSize

Constraints

OnCanResize

OnConstrainedResize

OnResize

UpdateBoundsRect

63.1.21.2.6 TRibbonGalleryControlMenuItem.SetContainedControl Method

Pascal

```
procedure SetContainedControl(const Value: TControl);
```

C++

```
__fastcall SetContainedControl(const TControl * Value);
```

Description

This is SetContainedControl, a member of class TRibbonGalleryControlMenuItem.

63.1.21.2.7 TRibbonGalleryControlMenuItem.SetParent Method

Sets the parent of the control.

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

SetParent is the protected implementation of the Parent property. Override SetParent to execute additional code when the value of the Parent property changes.

The AParent parameter specifies the new parent of the control.

If the control already has a parent, SetParent removes the control from that parent's list of controls by calling the parent's

RemoveControl method. If AParent is not nil (Delphi) or NULL (C++), SetParent calls its InsertControl method to add the control to its list of controls.

The TControl destructor calls SetParent(nil (Delphi) or NULL (C++)) to remove the control from its parent's control list before destroying the component.

See Also

Destroy

InsertControl

Parent

RemoveControl

63.1.21.3 TRibbonGalleryControlMenuItem Properties

The properties of the TRibbonGalleryControlMenuItem class are listed here.

Public Properties

	Name	Description
	ContainedControl (see page 3728)	This is ContainedControl, a member of class TRibbonGalleryControlMenuItem.

63.1.21.3.1 TRibbonGalleryControlMenuItem.ContainedControl Property

Pascal

```
property ContainedControl: TControl;
```

C++

```
__property TControl * ContainedControl;
```

Description

This is ContainedControl, a member of class TRibbonGalleryControlMenuItem.

63.1.22 TRibbonGalleryItem Class

Class Hierarchy

Pascal

```
TRibbonGalleryItem = class(TRibbonBaseButtonControl);
```

C++

```
class TRibbonGalleryItem : public TRibbonBaseButtonControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonGalleryItem.

63.1.22.1 TRibbonGalleryItem Members

The following tables list the members exposed by TRibbonGalleryItem.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	CalcLayout (see page 3730)	This is CalcLayout, a member of class TRibbonGalleryItem.
	DrawBackground (see page 3621)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawText (see page 3622)	Draws text on the control.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3730)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.
	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.22.2 TRibbonGalleryItem Methods

The methods of the TRibbonGalleryItem class are listed here.

Protected Methods

	Name	Description
	CalcLayout (see page 3730)	This is CalcLayout, a member of class TRibbonGalleryItem.

Public Methods

	Name	Description
	CalcBounds (see page 3730)	Calculates the size of the control.

63.1.22.2.1 TRibbonGalleryItem.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.22.2.2 TRibbonGalleryItem.CalcLayout Method

Pascal

```
procedure CalcLayout; override;
```

C++

```
virtual __fastcall CalcLayout();
```

Description

This is CalcLayout, a member of class TRibbonGalleryItem.

63.1.23 TRibbonGalleryMenuItem Class

Class Hierarchy

Pascal

```
TRibbonGalleryMenuItem = class(TCustomMenuItem);
```

C++

```
class TRibbonGalleryMenuItem : public TCustomMenuItem;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonGalleryMenuItem.

63.1.23.1 TRibbonGalleryMenuItem Members

The following tables list the members exposed by TRibbonGalleryMenuItem.

Protected Methods

	Name	Description
	DrawBackground (see page 3732)	Draws a background image on the control.
	DrawGlyph (see page 3732)	Draws a symbol on the control.
	DrawShadowedText (see page 3732)	This is DrawShadowedText, a member of class TRibbonGalleryMenuItem.
	DrawText (see page 3733)	Draws text on the control.

Public Methods

	Name	Description
	CalcBounds (see page 3731)	Calculates the size of the menu item.

63.1.23.2 TRibbonGalleryMenuItem Methods

The methods of the TRibbonGalleryMenuItem class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3732)	Draws a background image on the control.
	DrawGlyph (see page 3732)	Draws a symbol on the control.
	DrawShadowedText (see page 3732)	This is DrawShadowedText, a member of class TRibbonGalleryMenuItem.
	DrawText (see page 3733)	Draws text on the control.

Public Methods

	Name	Description
	CalcBounds (see page 3731)	Calculates the size of the menu item.

63.1.23.2.1 TRibbonGalleryMenuItem.CalcBounds Method

Calculates the size of the menu item.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as whether or not the menu item is a separator or not. CalcBounds computes the boundaries of the menu item considering these kinds of factors. At the end, it calls SetBounds and other routines

to ensure that the values are set properly.

See Also

SetBounds

63.1.23.2.2 TRibbonGalleryMenuItem.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.23.2.3 TRibbonGalleryMenuItem.DrawGlyph Method

Draws a symbol on the control.

Pascal

```
procedure DrawGlyph(const Location: TPoint); override;
```

C++

```
virtual __fastcall DrawGlyph(const TPoint Location);
```

Description

Use DrawGlyph to draw a symbol on the control at the specified location.

See Also

DrawBackground

DrawText

63.1.23.2.4 TRibbonGalleryMenuItem.DrawShadowedText Method

Pascal

```
procedure DrawShadowedText(Rect: TRect; Flags: Cardinal; Text: string; TextColor: TColor;
ShadowColor: TColor); override;
```

C++

```
virtual __fastcall DrawShadowedText(TRect Rect, unsigned Flags, UnicodeString Text, TColor
TextColor, TColor ShadowColor);
```

Description

This is DrawShadowedText, a member of class TRibbonGalleryMenuItem.

63.1.23.2.5 TRibbonGalleryMenuItem.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var ARect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect ARect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

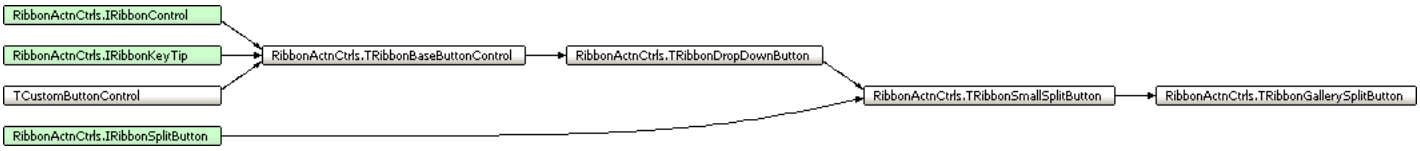
See Also

DrawBackground

DrawGlyph

63.1.24 TRibbonGallerySplitButton Class

Class Hierarchy



Pascal

```
TRibbonGallerySplitButton = class(TRibbonSmallSplitButton);
```

C++

```
class TRibbonGallerySplitButton : public TRibbonSmallSplitButton;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonGallerySplitButton.

63.1.24.1 TRibbonGallerySplitButton Members

The following tables list the members exposed by TRibbonGallerySplitButton.

Protected Fields

	Name	Description
	FCursorInSplit (see page 3791)	This is FCursorInSplit, a member of class TRibbonSmallSplitButton.
	FSplitRect (see page 3791)	This is FSplitRect, a member of class TRibbonSmallSplitButton.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	CalcLayout (see page 3717)	This is CalcLayout, a member of class TRibbonDropDownButton.
	CalcLayout (see page 3792)	This is CalcLayout, a member of class TRibbonSmallSplitButton.
	CMMouseLeave (see page 3793)	This is CMMouseLeave, a member of class TRibbonSmallSplitButton.
	DoDropDown (see page 3718)	This is DoDropDown, a member of class TRibbonDropDownButton.
	DrawArrow (see page 3718)	This is DrawArrow, a member of class TRibbonDropDownButton.
	DrawBackground (see page 3621)	Draws a background image on the control.
	DrawBackground (see page 3718)	Draws a background image on the control.
	DrawBackground (see page 3793)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3718)	This is DrawShadowedText, a member of class TRibbonDropDownButton.
	DrawText (see page 3622)	Draws text on the control.
	DrawText (see page 3719)	Draws text on the control.
	GetArrowPosition (see page 3719)	This is GetArrowPosition, a member of class TRibbonDropDownButton.
	GetArrowPosition (see page 3794)	This is GetArrowPosition, a member of class TRibbonSmallSplitButton.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	GetPopupClass (see page 3735)	This is GetPopupClass, a member of class TRibbonGallerySplitButton.
	InSplit (see page 3794)	This is InSplit, a member of class TRibbonSmallSplitButton.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	MouseDown (see page 3794)	OnMouseDown event dispatcher.
	MouseMove (see page 3795)	Respond to mouse moving over control area..
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3619)	Calculates the size of the control.
	CalcBounds (see page 3717)	Calculates the size of the control.
	CalcBounds (see page 3792)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.
	Click (see page 3717)	Executes the control's function.
	Click (see page 3793)	Executes the control's function.
	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Create (see page 3790)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.

	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
V	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
V	KeyTipClick (see page 3719)	Handle clicking a **KeyTip**.
V	KeyTipClick (see page 3794)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.
	UpdateSplitBounds (see page 3923)	This is UpdateSplitBounds, a member of class IRibbonSplitButton.
V	UpdateSplitBounds (see page 3795)	This is UpdateSplitBounds, a member of class TRibbonSmallSplitButton.

Public Properties

	Name	Description
	CursorInSplit (see page 3796)	This is CursorInSplit, a member of class TRibbonSmallSplitButton.
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.24.2 TRibbonGallerySplitButton Methods

The methods of the TRibbonGallerySplitButton class are listed here.

Protected Methods

	Name	Description
V	GetPopupClass (see page 3735)	This is GetPopupClass, a member of class TRibbonGallerySplitButton.

63.1.24.2.1 TRibbonGallerySplitButton.GetPopupClass Method

Pascal

```
function GetPopupClass: TCustomPopupClass; override;
```

C++

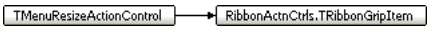
```
virtual __fastcall TCustomPopupClass GetPopupClass();
```

Description

This is GetPopupClass, a member of class TRibbonGallerySplitButton.

63.1.25 TRibbonGripItem Class

Class Hierarchy



Pascal

```
TRibbonGripItem = class(TMenuResizeActionControl);
```

C++

```
class TRibbonGripItem : public TMenuResizeActionControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonGripItem.

63.1.25.1 TRibbonGripItem Members

The following tables list the members exposed by TRibbonGripItem.

Protected Methods

	Name	Description
	Paint (see page 3737)	Render control surface.
	RibbonStyle (see page 3738)	This is RibbonStyle, a member of class TRibbonGripItem.

Public Methods

	Name	Description
	CalcBounds (see page 3737)	Calculates the size of the control.
	Create (see page 3736)	Creates an instance of TCustomActionControl.

Public Properties

	Name	Description
	GalleryResize (see page 3738)	This is GalleryResize, a member of class TRibbonGripItem.

63.1.25.2 TRibbonGripItem.Create Constructor

Creates an instance of TCustomActionControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonGripItem(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomActionControl. However, you should never attempt to instantiate a TCustomActionControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

AOwner is the component that is responsible for freeing the TCustomActionControl instance. It becomes the value of the Owner property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Owner

Create

Destroy

63.1.25.3 TRibbonGripItem Methods

The methods of the TRibbonGripItem class are listed here.

Protected Methods

	Name	Description
	Paint (see page 3737)	Render control surface.
	RibbonStyle (see page 3738)	This is RibbonStyle, a member of class TRibbonGripItem.

Public Methods

	Name	Description
	CalcBounds (see page 3737)	Calculates the size of the control.

63.1.25.3.1 TRibbonGripItem.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as the presence of an icon, caption, and shortcuts. CalcBounds computes the boundaries of the control considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.25.3.2 TRibbonGripItem.Paint Method

Render control surface.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method is called automatically when a lightweight control needs update its display area. Code in the Paint method uses the Canvas property to render the control surface.

As implemented in TGraphicControl, Paint does nothing. Specific rendering logic is provided by Paint methods in descendant control classes.

See Also

Canvas

63.1.25.3.3 TRibbonGripItem.RibbonStyle Method

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

This is RibbonStyle, a member of class TRibbonGripItem.

63.1.25.4 TRibbonGripItem Properties

The properties of the TRibbonGripItem class are listed here.

Public Properties

	Name	Description
	GalleryResize (see page 3738)	This is GalleryResize, a member of class TRibbonGripItem.

63.1.25.4.1 TRibbonGripItem.GalleryResize Property

Pascal

```
property GalleryResize: TGalleryResize;
```

C++

```
__property TGalleryResize GalleryResize;
```

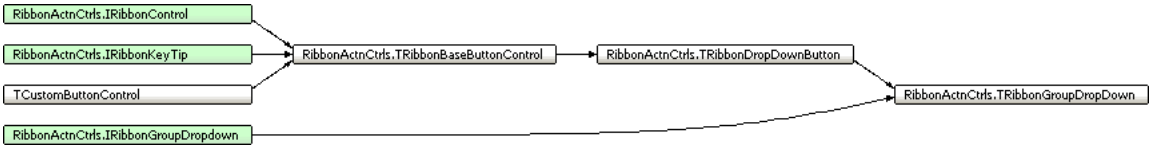
Description

This is GalleryResize, a member of class TRibbonGripItem.

63.1.26 TRibbonGroupDropDown Class

Display entire group as dropdown menu in Quick Access Toolbar.

Class Hierarchy



Pascal

```
TRibbonGroupDropDown = class(TRibbonDropDownButton, IRibbonGroupDropdown);
```

C++

```
class TRibbonGroupDropDown : public TRibbonDropDownButton, public IRibbonGroupDropdown;
```

File

RibbonActnCtrls

Description

TRibbonGroupDropDown is a control used to display an entire group as a dropdown menu in the Quick Access Toolbar.

See Also

TRibbonQuickAccessToolbar (see page 3546)

63.1.26.1 TRibbonGroupDropDown Members

The following tables list the members exposed by TRibbonGroupDropDown.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	CalcLayout (see page 3717)	This is CalcLayout, a member of class TRibbonDropDownButton.
	DoDropDown (see page 3740)	This is DoDropDown, a member of class TRibbonGroupDropDown.
	DrawArrow (see page 3741)	This is DrawArrow, a member of class TRibbonGroupDropDown.
	DrawBackground (see page 3741)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3718)	This is DrawShadowedText, a member of class TRibbonDropDownButton.
	DrawText (see page 3622)	Draws text on the control.
	DrawText (see page 3719)	Draws text on the control.
	GetArrowPosition (see page 3719)	This is GetArrowPosition, a member of class TRibbonDropDownButton.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	GetPopupClass (see page 3719)	This is GetPopupClass, a member of class TRibbonDropDownButton.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3619)	Calculates the size of the control.
	CalcBounds (see page 3717)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.
	Click (see page 3717)	Executes the control's function.
	CloseUp (see page 3740)	This is CloseUp, a member of class TRibbonGroupDropDown.
	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	GetParentControl (see page 3741)	This is GetParentControl, a member of class TRibbonGroupDropDown.
	GetTopPosition (see page 3741)	This is GetTopPosition, a member of class TRibbonGroupDropDown.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).

	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
V	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
V	KeyTipClick (see page 3719)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	Group (see page 3742)	This is Group, a member of class TRibbonGroupDropDown.
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.26.2 TRibbonGroupDropDown Methods

The methods of the TRibbonGroupDropDown class are listed here.

Protected Methods

	Name	Description
V	DoDropDown (see page 3740)	This is DoDropDown, a member of class TRibbonGroupDropDown.
V	DrawArrow (see page 3741)	This is DrawArrow, a member of class TRibbonGroupDropDown.
V	DrawBackground (see page 3741)	Draws a background image on the control.

Public Methods

	Name	Description
	CloseUp (see page 3740)	This is CloseUp, a member of class TRibbonGroupDropDown.
	GetParentControl (see page 3741)	This is GetParentControl, a member of class TRibbonGroupDropDown.
	GetTopPosition (see page 3741)	This is GetTopPosition, a member of class TRibbonGroupDropDown.

63.1.26.2.1 TRibbonGroupDropDown.CloseUp Method

Pascal

```
procedure CloseUp;
```

C++

```
__fastcall CloseUp();
```

Description

This is CloseUp, a member of class TRibbonGroupDropDown.

63.1.26.2.2 TRibbonGroupDropDown.DoDropDown Method

Pascal

```
procedure DoDropDown(const SelectFirst: Boolean); override;
```

C++

```
virtual __fastcall DoDropDown(const Boolean SelectFirst);
```

Description

This is DoDropDown, a member of class TRibbonGroupDropDown.

63.1.26.2.3 TRibbonGroupDropDown.DrawArrow Method

Pascal

```
procedure DrawArrow(Location: TPoint; Color: TColor; ShadowColor: TColor; Size: Integer);
override;
```

C++

```
virtual __fastcall DrawArrow(TPoint Location, TColor Color, TColor ShadowColor, int Size);
```

Description

This is DrawArrow, a member of class TRibbonGroupDropDown.

63.1.26.2.4 TRibbonGroupDropDown.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.26.2.5 TRibbonGroupDropDown.GetParentControl Method

Pascal

```
function GetParentControl: TWinControl;
```

C++

```
__fastcall TWinControl * GetParentControl();
```

Description

This is GetParentControl, a member of class TRibbonGroupDropDown.

63.1.26.2.6 TRibbonGroupDropDown.GetTopPosition Method

Pascal

```
function GetTopPosition: Integer;
```

C++

```
__fastcall int GetTopPosition();
```

Description

This is GetTopPosition, a member of class TRibbonGroupDropDown.

63.1.26.3 TRibbonGroupDropDown Properties

The properties of the TRibbonGroupDropDown class are listed here.

Public Properties

	Name	Description
	Group (see page 3742)	This is Group, a member of class TRibbonGroupDropDown.

63.1.26.3.1 TRibbonGroupDropDown.Group Property

Pascal

```
property Group: TControl;
```

C++

```
__property TControl * Group;
```

Description

This is Group, a member of class TRibbonGroupDropDown.

63.1.27 TRibbonLargeButtonControl Class

Class Hierarchy

Pascal

```
TRibbonLargeButtonControl = class(TRibbonBaseButtonControl);
```

C++

```
class TRibbonLargeButtonControl : public TRibbonBaseButtonControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonLargeButtonControl.

63.1.27.1 TRibbonLargeButtonControl Members

The following tables list the members exposed by TRibbonLargeButtonControl.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	CalcLayout (see page 3744)	This is CalcLayout, a member of class TRibbonLargeButtonControl.
	DrawBackground (see page 3744)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3745)	This is DrawShadowedText, a member of class TRibbonLargeButtonControl.
	DrawText (see page 3745)	Draws text on the control.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3744)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.
	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.27.2 TRibbonLargeButtonControl Methods

The methods of the TRibbonLargeButtonControl class are listed here.

Protected Methods

	Name	Description
	CalcLayout (see page 3744)	This is CalcLayout, a member of class TRibbonLargeButtonControl.

	DrawBackground (see page 3744)	Draws a background image on the control.
	DrawShadowedText (see page 3745)	This is DrawShadowedText, a member of class TRibbonLargeButtonControl.
	DrawText (see page 3745)	Draws text on the control.

Public Methods

	Name	Description
	CalcBounds (see page 3744)	Calculates the size of the control.

63.1.27.2.1 TRibbonLargeButtonControl.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.27.2.2 TRibbonLargeButtonControl.CalcLayout Method

Pascal

```
procedure CalcLayout; override;
```

C++

```
virtual __fastcall CalcLayout();
```

Description

This is CalcLayout, a member of class TRibbonLargeButtonControl.

63.1.27.2.3 TRibbonLargeButtonControl.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.27.2.4 TRibbonLargeButtonControl.DrawShadowedText Method

Pascal

```
procedure DrawShadowedText(Rect: TRect; Flags: Cardinal; Text: string; TextColor: TColor;
ShadowColor: TColor); override;
```

C++

```
virtual __fastcall DrawShadowedText(TRect Rect, unsigned Flags, UnicodeString Text, TColor
TextColor, TColor ShadowColor);
```

Description

This is DrawShadowedText, a member of class TRibbonLargeButtonControl.

63.1.27.2.5 TRibbonLargeButtonControl.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var ARect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect ARect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

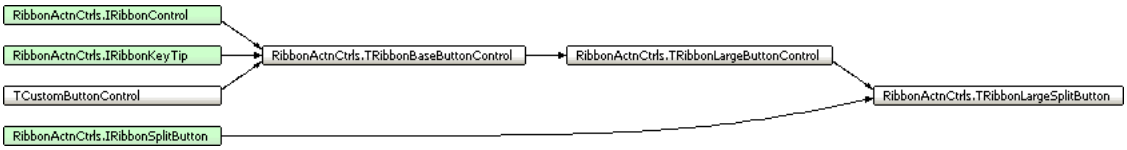
See Also

DrawBackground

DrawGlyph

63.1.28 TRibbonLargeSplitButton Class

Class Hierarchy



Pascal

```
TRibbonLargeSplitButton = class(TRibbonLargeButtonControl, IRibbonSplitButton);
```

C++

```
class TRibbonLargeSplitButton : public TRibbonLargeButtonControl, public IRibbonSplitButton;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonLargeSplitButton.

63.1.28.1 TRibbonLargeSplitButton Members

The following tables list the members exposed by TRibbonLargeSplitButton.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	CalcLayout (see page 3748)	This is CalcLayout, a member of class TRibbonLargeSplitButton.
	CMMouseleave (see page 3748)	This is CMMouseleave, a member of class TRibbonLargeSplitButton.
	DoSplitButtonClick (see page 3748)	This is DoSplitButtonClick, a member of class TRibbonLargeSplitButton.
	DrawArrow (see page 3749)	This is DrawArrow, a member of class TRibbonLargeSplitButton.
	DrawBackground (see page 3749)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3745)	This is DrawShadowedText, a member of class TRibbonLargeButtonControl.
	DrawText (see page 3622)	Draws text on the control.
	DrawText (see page 3745)	Draws text on the control.
	GetArrowPosition (see page 3749)	This is GetArrowPosition, a member of class TRibbonLargeSplitButton.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	InSplit (see page 3749)	This is InSplit, a member of class TRibbonLargeSplitButton.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	MouseDown (see page 3750)	OnMouseDown event dispatcher.
	MouseMove (see page 3750)	Respond to mouse moving over control area..
	MouseUp (see page 3751)	OnMouseUp event dispatcher.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3747)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3748)	Executes the control's function.
	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3750)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.

	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.
	UpdateSplitBounds (see page 3751)	This is UpdateSplitBounds, a member of class TRibbonLargeSplitButton.

Public Properties

	Name	Description
	CursorInSplit (see page 3752)	This is CursorInSplit, a member of class TRibbonLargeSplitButton.
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.28.2 TRibbonLargeSplitButton Methods

The methods of the TRibbonLargeSplitButton class are listed here.

Protected Methods

	Name	Description
	CalcLayout (see page 3748)	This is CalcLayout, a member of class TRibbonLargeSplitButton.
	CMMouseleave (see page 3748)	This is CMMouseleave, a member of class TRibbonLargeSplitButton.
	DoSplitButtonClick (see page 3748)	This is DoSplitButtonClick, a member of class TRibbonLargeSplitButton.
	DrawArrow (see page 3749)	This is DrawArrow, a member of class TRibbonLargeSplitButton.
	DrawBackground (see page 3749)	Draws a background image on the control.
	GetArrowPosition (see page 3749)	This is GetArrowPosition, a member of class TRibbonLargeSplitButton.
	InSplit (see page 3749)	This is InSplit, a member of class TRibbonLargeSplitButton.
	MouseDown (see page 3750)	OnMouseDown event dispatcher.
	MouseMove (see page 3750)	Respond to mouse moving over control area..
	MouseUp (see page 3751)	OnMouseUp event dispatcher.

Public Methods

	Name	Description
	CalcBounds (see page 3747)	Calculates the size of the control.
	Click (see page 3748)	Executes the control's function.
	KeyTipClick (see page 3750)	Handle clicking a **KeyTip**.
	UpdateSplitBounds (see page 3751)	This is UpdateSplitBounds, a member of class TRibbonLargeSplitButton.

63.1.28.2.1 TRibbonLargeSplitButton.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.28.2.2 TRibbonLargeSplitButton.CalcLayout Method

Pascal

```
procedure CalcLayout; override;
```

C++

```
virtual __fastcall CalcLayout();
```

Description

This is CalcLayout, a member of class TRibbonLargeSplitButton.

63.1.28.2.3 TRibbonLargeSplitButton.Click Method

Executes the control's function.

Pascal

```
procedure Click; override;
```

C++

```
virtual __fastcall Click();
```

Description

This override of the method inherited from TCustomActionControl is almost identical to the inherited method. The only difference is that if the button control is being manipulated at design time, extra actions are taken to ensure that the control is properly selected.

See Also

Click

Click

UsageCount

63.1.28.2.4 TRibbonLargeSplitButton.CMMouseleave Method

Pascal

```
procedure CMMouseleave(var Message: TMessage); message CM_MOUSELEAVE;
```

C++

```
__fastcall CMMouseleave(TMessage Message);
```

Description

This is CMMouseleave, a member of class TRibbonLargeSplitButton.

63.1.28.2.5 TRibbonLargeSplitButton.DoSplitButtonClick Method

Pascal

```
procedure DoSplitButtonClick(const SelectFirst: Boolean); virtual;
```

C++

```
virtual __fastcall DoSplitButtonClick(const Boolean SelectFirst);
```

Description

This is DoSplitButtonClick, a member of class TRibbonLargeSplitButton.

63.1.28.2.6 TRibbonLargeSplitButton.DrawArrow Method

Pascal

```
procedure DrawArrow(Location: TPoint; Color: TColor; ShadowColor: TColor; Size: Integer);
virtual;
```

C++

```
virtual __fastcall DrawArrow(TPoint Location, TColor Color, TColor ShadowColor, int Size);
```

Description

This is DrawArrow, a member of class TRibbonLargeSplitButton.

63.1.28.2.7 TRibbonLargeSplitButton.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.28.2.8 TRibbonLargeSplitButton.GetArrowPosition Method

Pascal

```
function GetArrowPosition(const ARect: TRect): TPoint; virtual;
```

C++

```
virtual __fastcall TPoint GetArrowPosition(const TRect ARect);
```

Description

This is GetArrowPosition, a member of class TRibbonLargeSplitButton.

63.1.28.2.9 TRibbonLargeSplitButton.InSplit Method

Pascal

```
function InSplit(X: Integer; Y: Integer): Boolean;
```

C++

```
__fastcall Boolean InSplit(int X, int Y);
```

Description

This is InSplit, a member of class TRibbonLargeSplitButton.

63.1.28.2.10 TRibbonLargeSplitButton.KeyTipClick Method

Handle clicking a **KeyTip**.

Pascal

```
procedure KeyTipClick; override;
```

C++

```
virtual __fastcall KeyTipClick();
```

Description

KeyTipClick Handle clicking a **KeyTip**. Some requires need to be handled differently when clicked by a **KeyTip**.

63.1.28.2.11 TRibbonLargeSplitButton.MouseDown Method

OnMouseDown event dispatcher.

Pascal

```
procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer);
override;
```

C++

```
virtual __fastcall MouseDown(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user presses the mouse button down while the cursor's hotspot is over the control.

The Button parameter determines which mouse button the user pressed. Shift indicates which shift keys (Shift, Ctrl, or Alt) were down when the user pressed the mouse button. X and Y are the pixel coordinates of the mouse pointer within the client area of the control.

A control calls MouseDown in response to any of the Windows mouse-down messages (WM_LBUTTONDOWN, WM_MBUTTONDOWN, WM_RBUTTONDOWN), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was pressed down: left, right, or middle.

See Also

MouseMove

MouseUp

OnMouseDown

63.1.28.2.12 TRibbonLargeSplitButton.MouseMove Method

Respond to mouse moving over control area..

Pascal

```
procedure MouseMove(Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseMove(TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseMove method to provide other responses in addition to calling the OnMouseMove event handler when the user moves the mouse.

A control calls MouseMove in response to any of the Windows mouse-move messages (WM_MOUSEMOVE), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively.

As the mouse cursor moves across a control, this method is called repeatedly. Each time it is called, it is with the new coordinates that reflect the continuous path of the mouse cursor across the screen real estate covered by the control's visual representation.

See Also

MouseDown

MouseUp

OnMouseMove

63.1.28.2.13 TRibbonLargeSplitButton.MouseUp Method

OnMouseUp event dispatcher.

Pascal

```
procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseUp(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user releases a previously pressed mouse button while the cursor's hotspot is over the control.

A control calls MouseUp in response to any of the Windows mouse-up messages (WM_LBUTTONUP, WM_MBUTTONUP, WM_RBUTTONUP), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was released: left, right, or middle

See Also

MouseDown

MouseMove

OnMouseUp

63.1.28.2.14 TRibbonLargeSplitButton.UpdateSplitBounds Method

Pascal

```
procedure UpdateSplitBounds; virtual;
```

C++

```
virtual __fastcall UpdateSplitBounds();
```

Description

This is UpdateSplitBounds, a member of class TRibbonLargeSplitButton.

63.1.28.3 TRibbonLargeSplitButton Properties

The properties of the TRibbonLargeSplitButton class are listed here.

Public Properties

	Name	Description
	CursorInSplit (see page 3752)	This is CursorInSplit, a member of class TRibbonLargeSplitButton.

63.1.28.3.1 TRibbonLargeSplitButton.CursorInSplit Property

Pascal

```
property CursorInSplit: Boolean;
```

C++

```
__property Boolean CursorInSplit;
```

Description

This is CursorInSplit, a member of class TRibbonLargeSplitButton.

63.1.29 TRibbonMainMenuItem Class

Class Hierarchy

Pascal

```
TRibbonMainMenuItem = class(TRibbonApplicationMenuItem);
```

C++

```
class TRibbonMainMenuItem : public TRibbonApplicationMenuItem;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonMainMenuItem.

63.1.29.1 TRibbonMainMenuItem Members

The following tables list the members exposed by TRibbonMainMenuItem.

Protected Methods

	Name	Description
	CalcLayout (see page 3757)	This is CalcLayout, a member of class TRibbonMenuItem.
	DrawBackground (see page 3758)	Draws a background image on the control.

	DrawGlyph (see page 3614)	Draws a symbol on the control.
	DrawGlyph (see page 3758)	Draws a symbol on the control.
	DrawSelectedBackground (see page 3615)	This is DrawSelectedBackground, a member of class TRibbonApplicationMenuItem.
	DrawSelectedBackground (see page 3758)	This is DrawSelectedBackground, a member of class TRibbonMenuItem.
	DrawSeparator (see page 3615)	This is DrawSeparator, a member of class TRibbonApplicationMenuItem.
	DrawSeparator (see page 3759)	This is DrawSeparator, a member of class TRibbonMenuItem.
	DrawShadowedText (see page 3753)	This is DrawShadowedText, a member of class TRibbonMainMenuItem.
	DrawSubMenuGlyph (see page 3615)	This is DrawSubMenuGlyph, a member of class TRibbonApplicationMenuItem.
	DrawText (see page 3753)	Draws text on the control.
	GetImageSize (see page 3616)	This is GetImageSize, a member of class TRibbonApplicationMenuItem.
	GetShowShortCut (see page 3759)	This is GetShowShortCut, a member of class TRibbonMenuItem.
	RibbonStyle (see page 3760)	This is RibbonStyle, a member of class TRibbonMenuItem.

Protected Properties

	Name	Description
	DrawBanner (see page 3760)	This is DrawBanner, a member of class TRibbonMenuItem.

Public Methods

	Name	Description
	CalcBounds (see page 3614)	Calculates the size of the menu item.
	CalcBounds (see page 3757)	Calculates the size of the menu item.
	Click (see page 3757)	Executes the control's function.
	Create (see page 3613)	Creates an instance of TCustomMenuItem.
	Create (see page 3756)	Creates an instance of TCustomMenuItem.

63.1.29.2 TRibbonMainMenuItem Methods

The methods of the TRibbonMainMenuItem class are listed here.

Protected Methods

	Name	Description
	DrawShadowedText (see page 3753)	This is DrawShadowedText, a member of class TRibbonMainMenuItem.
	DrawText (see page 3753)	Draws text on the control.

63.1.29.2.1 TRibbonMainMenuItem.DrawShadowedText Method

Pascal

```
procedure DrawShadowedText(Rect: TRect; Flags: Cardinal; Text: string; TextColor: TColor;
ShadowColor: TColor); override;
```

C++

```
virtual __fastcall DrawShadowedText(TRect Rect, unsigned Flags, UnicodeString Text, TColor
TextColor, TColor ShadowColor);
```

Description

This is DrawShadowedText, a member of class TRibbonMainMenuItem.

63.1.29.2.2 TRibbonMainMenuItem.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var ARect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect ARect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

See Also

DrawBackground

DrawGlyph

63.1.30 TRibbonMainMenuSeparator Class

Class Hierarchy

Pascal

```
TRibbonMainMenuSeparator = class(TRibbonSeparator);
```

C++

```
class TRibbonMainMenuSeparator : public TRibbonSeparator;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonMainMenuSeparator.

63.1.30.1 TRibbonMainMenuSeparator Members

The following tables list the members exposed by TRibbonMainMenuSeparator.

Protected Methods

	Name	Description
	DrawBackground (see page 3785)	Draws a background image on the control.
	DrawGlyph (see page 3786)	Draws a symbol on the control.
	DrawShadowedText (see page 3786)	This is DrawShadowedText, a member of class TRibbonSeparator.
	DrawText (see page 3786)	Draws text on the control.
	RibbonStyle (see page 3787)	This is RibbonStyle, a member of class TRibbonSeparator.

Public Methods

	Name	Description
	CalcBounds (see page 3755)	Calculates the size of the control.
	Create (see page 3784)	Creates an instance of TCustomActionControl.

63.1.30.2 TRibbonMainMenuSeparator Methods

The methods of the TRibbonMainMenuSeparator class are listed here.

Public Methods

	Name	Description
	CalcBounds (see page 3755)	Calculates the size of the control.

63.1.30.2.1 TRibbonMainMenuSeparator.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as the presence of an icon, caption, and shortcuts. CalcBounds computes the boundaries of the control considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.31 TRibbonMenuItem Class

Class Hierarchy

Pascal

```
TRibbonMenuItem = class(TCustomMenuItem);
```

C++

```
class TRibbonMenuItem : public TCustomMenuItem;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonMenuItem.

63.1.31.1 TRibbonMenuItem Members

The following tables list the members exposed by TRibbonMenuItem.

Protected Methods

	Name	Description
	CalcLayout (see page 3757)	This is CalcLayout, a member of class TRibbonMenuItem.
	DrawBackground (see page 3758)	Draws a background image on the control.
	DrawGlyph (see page 3758)	Draws a symbol on the control.
	DrawSelectedBackground (see page 3758)	This is DrawSelectedBackground, a member of class TRibbonMenuItem.
	DrawSeparator (see page 3759)	This is DrawSeparator, a member of class TRibbonMenuItem.
	DrawShadowedText (see page 3759)	This is DrawShadowedText, a member of class TRibbonMenuItem.
	DrawText (see page 3759)	Draws text on the control.
	GetShowShortCut (see page 3759)	This is GetShowShortCut, a member of class TRibbonMenuItem.
	RibbonStyle (see page 3760)	This is RibbonStyle, a member of class TRibbonMenuItem.

Protected Properties

	Name	Description
	DrawBanner (see page 3760)	This is DrawBanner, a member of class TRibbonMenuItem.

Public Methods

	Name	Description
	CalcBounds (see page 3757)	Calculates the size of the menu item.
	Click (see page 3757)	Executes the control's function.
	Create (see page 3756)	Creates an instance of TCustomMenuItem.

63.1.31.2 TRibbonMenuItem.Create Constructor

Creates an instance of TCustomMenuItem.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonMenuItem(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomMenuItem. However, you should never attempt to instantiate a TCustomMenuItem. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

After calling the inherited constructor, Create initializes the control, including the following property settings:

Initializes the TCanvas object for its Canvas property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Create

Destroy

63.1.31.3 TRibbonMenuItem Methods

The methods of the TRibbonMenuItem class are listed here.

Protected Methods

	Name	Description
	CalcLayout (see page 3757)	This is CalcLayout, a member of class TRibbonMenuItem.
	DrawBackground (see page 3758)	Draws a background image on the control.
	DrawGlyph (see page 3758)	Draws a symbol on the control.
	DrawSelectedBackground (see page 3758)	This is DrawSelectedBackground, a member of class TRibbonMenuItem.
	DrawSeparator (see page 3759)	This is DrawSeparator, a member of class TRibbonMenuItem.
	DrawShadowedText (see page 3759)	This is DrawShadowedText, a member of class TRibbonMenuItem.
	DrawText (see page 3759)	Draws text on the control.
	GetShowShortCut (see page 3759)	This is GetShowShortCut, a member of class TRibbonMenuItem.
	RibbonStyle (see page 3760)	This is RibbonStyle, a member of class TRibbonMenuItem.

Public Methods

	Name	Description
	CalcBounds (see page 3757)	Calculates the size of the menu item.
	Click (see page 3757)	Executes the control's function.

63.1.31.3.1 TRibbonMenuItem.CalcBounds Method

Calculates the size of the menu item.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as whether or not the menu item is a separator or not. CalcBounds computes the boundaries of the menu item considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.31.3.2 TRibbonMenuItem.CalcLayout Method

Pascal

```
procedure CalcLayout; override;
```

C++

```
virtual __fastcall CalcLayout();
```

Description

This is CalcLayout, a member of class TRibbonMenuItem.

63.1.31.3.3 TRibbonMenuItem.Click Method

Executes the control's function.

Pascal

```
procedure Click; override;
```

C++

```
virtual __fastcall Click();
```

Description

This override of the method inherited from TCustomActionControl does little except bring the form containing the item into focus, and prevent the inherited method from being called. The reason for this lack of action is that at runtime, no action should be taken until the primary mouse button is pressed and released, which makes the OnClick event handler responsible for taking action. Click takes place only when the mouse button is first pressed, prior to release.

See Also

OnClick

63.1.31.3.4 TRibbonMenuItem.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.31.3.5 TRibbonMenuItem.DrawGlyph Method

Draws a symbol on the control.

Pascal

```
procedure DrawGlyph(const Location: TPoint); override;
```

C++

```
virtual __fastcall DrawGlyph(const TPoint Location);
```

Description

Use DrawGlyph to draw a symbol on the control at the specified location.

See Also

DrawBackground

DrawText

63.1.31.3.6 TRibbonMenuItem.DrawSelectedBackground Method

Pascal

```
procedure DrawSelectedBackground(ARect: TRect); virtual;
```

C++

```
virtual __fastcall DrawSelectedBackground(TRect ARect);
```

Description

This is DrawSelectedBackground, a member of class TRibbonMenuItem.

63.1.31.3.7 TRibbonMenuItem.DrawSeparator Method

Pascal

```
procedure DrawSeparator(const Offset: Integer); override;
```

C++

```
virtual __fastcall DrawSeparator(const int Offset);
```

Description

This is DrawSeparator, a member of class TRibbonMenuItem.

63.1.31.3.8 TRibbonMenuItem.DrawShadowedText Method

Pascal

```
procedure DrawShadowedText(Rect: TRect; Flags: Cardinal; Text: string; TextColor: TColor;
ShadowColor: TColor); override;
```

C++

```
virtual __fastcall DrawShadowedText(TRect Rect, unsigned Flags, UnicodeString Text, TColor
TextColor, TColor ShadowColor);
```

Description

This is DrawShadowedText, a member of class TRibbonMenuItem.

63.1.31.3.9 TRibbonMenuItem.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var ARect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect ARect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

See Also

DrawBackground

DrawGlyph

63.1.31.3.10 TRibbonMenuItem.GetShowShortCut Method

Pascal

```
function GetShowShortCut: Boolean; override;
```

C++

```
virtual __fastcall Boolean GetShowShortCut();
```

Description

This is GetShowShortCut, a member of class TRibbonMenuItem.

63.1.31.3.11 TRibbonMenuItem.RibbonStyle Method

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

This is RibbonStyle, a member of class TRibbonMenuItem.

63.1.31.4 TRibbonMenuItem Properties

The properties of the TRibbonMenuItem class are listed here.

Protected Properties

	Name	Description
	DrawBanner (see page 3760)	This is DrawBanner, a member of class TRibbonMenuItem.

63.1.31.4.1 TRibbonMenuItem.DrawBanner Property

Pascal

```
property DrawBanner: Boolean;
```

C++

```
__property Boolean DrawBanner;
```

Description

This is DrawBanner, a member of class TRibbonMenuItem.

63.1.32 TRibbonMenuSeparatorItem Class

Class Hierarchy

Pascal

```
TRibbonMenuSeparatorItem = class(TRibbonMenuItem);
```

C++

```
class TRibbonMenuSeparatorItem : public TRibbonMenuItem;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonMenuSeparatorItem.

63.1.32.1 TRibbonMenuSeparatorItem Members

The following tables list the members exposed by TRibbonMenuSeparatorItem.

Protected Methods

	Name	Description
	CalcLayout (see page 3757)	This is CalcLayout, a member of class TRibbonMenuItem.
	DrawBackground (see page 3758)	Draws a background image on the control.
	DrawGlyph (see page 3758)	Draws a symbol on the control.
	DrawSelectedBackground (see page 3758)	This is DrawSelectedBackground, a member of class TRibbonMenuItem.
	DrawSeparator (see page 3762)	This is DrawSeparator, a member of class TRibbonMenuSeparatorItem.
	DrawShadowedText (see page 3759)	This is DrawShadowedText, a member of class TRibbonMenuItem.
	DrawText (see page 3759)	Draws text on the control.
	GetShowShortCut (see page 3759)	This is GetShowShortCut, a member of class TRibbonMenuItem.
	RibbonStyle (see page 3760)	This is RibbonStyle, a member of class TRibbonMenuItem.

Protected Properties

	Name	Description
	DrawBanner (see page 3760)	This is DrawBanner, a member of class TRibbonMenuItem.

Public Methods

	Name	Description
	CalcBounds (see page 3761)	Calculates the size of the menu item.
	Click (see page 3757)	Executes the control's function.
	Create (see page 3756)	Creates an instance of TCustomMenuItem.

63.1.32.2 TRibbonMenuSeparatorItem Methods

The methods of the TRibbonMenuSeparatorItem class are listed here.

Protected Methods

	Name	Description
	DrawSeparator (see page 3762)	This is DrawSeparator, a member of class TRibbonMenuSeparatorItem.

Public Methods

	Name	Description
	CalcBounds (see page 3761)	Calculates the size of the menu item.

63.1.32.2.1 TRibbonMenuSeparatorItem.CalcBounds Method

Calculates the size of the menu item.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as whether or not the menu item is a separator or not. CalcBounds computes the boundaries of the menu item considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.32.2.2 TRibbonMenuSeparatorItem.DrawSeparator Method

Pascal

```
procedure DrawSeparator(const Offset: Integer); override;
```

C++

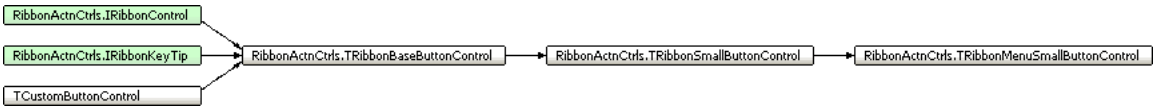
```
virtual __fastcall DrawSeparator(const int Offset);
```

Description

This is DrawSeparator, a member of class TRibbonMenuSeparatorItem.

63.1.33 TRibbonMenuSmallButtonControl Class

Class Hierarchy



Pascal

```
TRibbonMenuSmallButtonControl = class(TRibbonSmallButtonControl);
```

C++

```
class TRibbonMenuSmallButtonControl : public TRibbonSmallButtonControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonMenuSmallButtonControl.

63.1.33.1 TRibbonMenuSmallButtonControl Members

The following tables list the members exposed by TRibbonMenuSmallButtonControl.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.

	DrawBackground (see page 3621)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawText (see page 3622)	Draws text on the control.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	MouseUp (see page 3764)	OnMouseUp event dispatcher.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3619)	Calculates the size of the control.
	CalcBounds (see page 3788)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.
	Create (see page 3763)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.33.2 TRibbonMenuSmallButtonControl.Create Constructor

Creates an instance of TCustomButtonControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonMenuSmallButtonControl(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomButtonControl. However, you should never attempt to instantiate a TCustomButtonControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

AOwner is the component that is responsible for freeing the button control. It becomes the value of the Owner property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Owner

Create

Destroy

63.1.33.3 TRibbonMenuSmallButtonControl Methods

The methods of the TRibbonMenuSmallButtonControl class are listed here.

Protected Methods

	Name	Description
	MouseUp (see page 3764)	OnMouseUp event dispatcher.

63.1.33.3.1 TRibbonMenuSmallButtonControl.MouseUp Method

OnMouseUp event dispatcher.

Pascal

```
procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseUp(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user releases a previously pressed mouse button while the cursor's hotspot is over the control.

A control calls MouseUp in response to any of the Windows mouse-up messages (WM_LBUTTONUP, WM_MBUTTONUP, WM_RBUTTONUP), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was released: left, right, or middle

See Also

MouseDown

MouseMove

OnMouseUp

63.1.34 TRibbonOptionButtonControl Class

Class Hierarchy

Pascal

```
TRibbonOptionButtonControl = class(TRibbonMenuSmallButtonControl);
```

C++

```
class TRibbonOptionButtonControl : public TRibbonMenuSmallButtonControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonOptionButtonControl.

63.1.34.1 TRibbonOptionButtonControl Members

The following tables list the members exposed by TRibbonOptionButtonControl.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	DrawBackground (see page 3766)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawText (see page 3622)	Draws text on the control.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	MouseUp (see page 3764)	OnMouseUp event dispatcher.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3619)	Calculates the size of the control.
	CalcBounds (see page 3788)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.

	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Create (see page 3763)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.34.2 TRibbonOptionButtonControl Methods

The methods of the TRibbonOptionButtonControl class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3766)	Draws a background image on the control.

63.1.34.2.1 TRibbonOptionButtonControl.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.35 TRibbonQATScrollButton Class

Class Hierarchy

Pascal

```
TRibbonQATScrollButton = class(TCustomToolScrollBtn);
```

C++

```
class TRibbonQATScrollButton : public TCustomToolScrollBtn;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonQATScrollButton.

63.1.35.1 TRibbonQATScrollButton Members

The following tables list the members exposed by TRibbonQATScrollButton.

Protected Methods

	Name	Description
	CMMouseEnter (see page 3769)	This is CMMouseEnter, a member of class TRibbonQATScrollButton.
	CMMouseLeave (see page 3769)	This is CMMouseLeave, a member of class TRibbonQATScrollButton.
	DrawArrows (see page 3769)	This is DrawArrows, a member of class TRibbonQATScrollButton.
	DrawBackground (see page 3769)	Draws a background image on the control.
	MouseMove (see page 3770)	Respond to mouse moving over control area..
	RibbonStyle (see page 3770)	This is RibbonStyle, a member of class TRibbonQATScrollButton.

Protected Properties

	Name	Description
	CursorInBtn (see page 3771)	This is CursorInBtn, a member of class TRibbonQATScrollButton.

Public Methods

	Name	Description
	CalcBounds (see page 3768)	Calculates the size of the control.
	Click (see page 3768)	Executes the control's function.

Public Properties

	Name	Description
	DrawDown (see page 3771)	This is DrawDown, a member of class TRibbonQATScrollButton.
	ScrollMode (see page 3771)	This is ScrollMode, a member of class TRibbonQATScrollButton.

63.1.35.2 TRibbonQATScrollButton Methods

The methods of the TRibbonQATScrollButton class are listed here.

Protected Methods

	Name	Description
	CMMouseEnter (see page 3769)	This is CMMouseEnter, a member of class TRibbonQATScrollButton.
	CMMouseLeave (see page 3769)	This is CMMouseLeave, a member of class TRibbonQATScrollButton.
	DrawArrows (see page 3769)	This is DrawArrows, a member of class TRibbonQATScrollButton.
	DrawBackground (see page 3769)	Draws a background image on the control.
	MouseMove (see page 3770)	Respond to mouse moving over control area..
	RibbonStyle (see page 3770)	This is RibbonStyle, a member of class TRibbonQATScrollButton.

Public Methods

	Name	Description
	CalcBounds (see page 3768)	Calculates the size of the control.
	Click (see page 3768)	Executes the control's function.

63.1.35.2.1 TRibbonQATScrollButton.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.35.2.2 TRibbonQATScrollButton.Click Method

Executes the control's function.

Pascal

```
procedure Click; override;
```

C++

```
virtual __fastcall Click();
```

Description

This override of the method inherited from TCustomActionControl is almost identical to the inherited method. The only difference is that if the button control is being manipulated at design time, extra actions are taken to ensure that the control is properly selected.

See Also

Click

Click

UsageCount

63.1.35.2.3 TRibbonQATScrollButton.CMMouseEnter Method

Pascal

```
procedure CMMouseEnter(var Message: TMessage); override;
```

C++

```
virtual __fastcall CMMouseEnter(TMessage Message);
```

Description

This is CMMouseEnter, a member of class TRibbonQATScrollButton.

63.1.35.2.4 TRibbonQATScrollButton.CMMouseLeave Method

Pascal

```
procedure CMMouseLeave(var Message: TMessage); override;
```

C++

```
virtual __fastcall CMMouseLeave(TMessage Message);
```

Description

This is CMMouseLeave, a member of class TRibbonQATScrollButton.

63.1.35.2.5 TRibbonQATScrollButton.DrawArrows Method

Pascal

```
procedure DrawArrows; override;
```

C++

```
virtual __fastcall DrawArrows();
```

Description

This is DrawArrows, a member of class TRibbonQATScrollButton.

63.1.35.2.6 TRibbonQATScrollButton.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.35.2.7 TRibbonQATScrollButton.MouseMove Method

Respond to mouse moving over control area..

Pascal

```
procedure MouseMove(Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseMove(TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseMove method to provide other responses in addition to calling the OnMouseMove event handler when the user moves the mouse.

A control calls MouseMove in response to any of the Windows mouse-move messages (WM_MOUSEMOVE), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively.

As the mouse cursor moves across a control, this method is called repeatedly. Each time it is called, it is with the new coordinates that reflect the continuous path of the mouse cursor across the screen real estate covered by the control's visual representation.

See Also

MouseDown

MouseUp

OnMouseMove

63.1.35.2.8 TRibbonQATScrollButton.RibbonStyle Method

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

This is RibbonStyle, a member of class TRibbonQATScrollButton.

63.1.35.3 TRibbonQATScrollButton Properties

The properties of the TRibbonQATScrollButton class are listed here.

Protected Properties

	Name	Description
	CursorInBtn (see page 3771)	This is CursorInBtn, a member of class TRibbonQATScrollButton.

Public Properties

	Name	Description
	DrawDown (see page 3771)	This is DrawDown, a member of class TRibbonQATScrollButton.
	ScrollMode (see page 3771)	This is ScrollMode, a member of class TRibbonQATScrollButton.

63.1.35.3.1 TRibbonQATScrollButton.CursorInBtn Property

Pascal

```
property CursorInBtn: Boolean;
```

C++

```
__property Boolean CursorInBtn;
```

Description

This is CursorInBtn, a member of class TRibbonQATScrollButton.

63.1.35.3.2 TRibbonQATScrollButton.DrawDown Property

Pascal

```
property DrawDown: Boolean;
```

C++

```
__property Boolean DrawDown;
```

Description

This is DrawDown, a member of class TRibbonQATScrollButton.

63.1.35.3.3 TRibbonQATScrollButton.ScrollMode Property

Pascal

```
property ScrollMode: TQATScrollMode;
```

C++

```
__property TQATScrollMode ScrollMode;
```

Description

This is ScrollMode, a member of class TRibbonQATScrollButton.

63.1.36 TRibbonRadioButtonControl Class

Class Hierarchy

Pascal

```
TRibbonRadioButtonControl = class(TRibbonSmallButtonControl);
```

C++

```
class TRibbonRadioButtonControl : public TRibbonSmallButtonControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonRadioButtonControl.

63.1.36.1 TRibbonRadioButtonControl Members

The following tables list the members exposed by TRibbonRadioButtonControl.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	DrawBackground (see page 3773)	Draws a background image on the control.
	DrawGlyph (see page 3773)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawText (see page 3622)	Draws text on the control.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3773)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.
	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.36.2 TRibbonRadioButtonControl Methods

The methods of the TRibbonRadioButtonControl class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3773)	Draws a background image on the control.
	DrawGlyph (see page 3773)	Draws a symbol on the control.

Public Methods

	Name	Description
	CalcBounds (see page 3773)	Calculates the size of the control.

63.1.36.2.1 TRibbonRadioButtonControl.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.36.2.2 TRibbonRadioButtonControl.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.36.2.3 TRibbonRadioButtonControl.DrawGlyph Method

Draws a symbol on the control.

Pascal

```
procedure DrawGlyph(const Location: TPoint); override;
```

C++

```
virtual __fastcall DrawGlyph(const TPoint Location);
```

Description

Use DrawGlyph to draw a symbol on the control at the specified location.

See Also

DrawBackground

DrawText

63.1.37 TRibbonRecentButtonControl Class

Class Hierarchy

Pascal

```
TRibbonRecentButtonControl = class(TRibbonMenuSmallButtonControl);
```

C++

```
class TRibbonRecentButtonControl : public TRibbonMenuSmallButtonControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonRecentButtonControl.

63.1.37.1 TRibbonRecentButtonControl Members

The following tables list the members exposed by TRibbonRecentButtonControl.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3775)	This is AllowHotKeys, a member of class TRibbonRecentButtonControl.
	DrawBackground (see page 3621)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawText (see page 3622)	Draws text on the control.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	MouseUp (see page 3764)	OnMouseUp event dispatcher.

	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3619)	Calculates the size of the control.
	CalcBounds (see page 3788)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.
	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Create (see page 3763)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.37.2 TRibbonRecentButtonControl Methods

The methods of the TRibbonRecentButtonControl class are listed here.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3775)	This is AllowHotKeys, a member of class TRibbonRecentButtonControl.

63.1.37.2.1 TRibbonRecentButtonControl.AllowHotKeys Method

Pascal

```
function AllowHotKeys: Boolean; override;
```

C++

```
virtual __fastcall Boolean AllowHotKeys();
```

Description

This is AllowHotKeys, a member of class TRibbonRecentButtonControl.

63.1.38 TRibbonRichContentItem Class

Class Hierarchy

Pascal

```
TRibbonRichContentItem = class(TRibbonMenuItem);
```

C++

```
class TRibbonRichContentItem : public TRibbonMenuItem;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonRichContentItem.

63.1.38.1 TRibbonRichContentItem Members

The following tables list the members exposed by TRibbonRichContentItem.

Protected Methods

	Name	Description
	CalcLayout (see page 3757)	This is CalcLayout, a member of class TRibbonMenuItem.
	CanCalculateHeight (see page 3778)	This is CanCalculateHeight, a member of class TRibbonRichContentItem.
	DrawBackground (see page 3778)	Draws a background image on the control.
	DrawGlyph (see page 3778)	Draws a symbol on the control.
	DrawSelectedBackground (see page 3758)	This is DrawSelectedBackground, a member of class TRibbonMenuItem.
	DrawSeparator (see page 3759)	This is DrawSeparator, a member of class TRibbonMenuItem.
	DrawShadowedText (see page 3779)	This is DrawShadowedText, a member of class TRibbonRichContentItem.
	DrawText (see page 3779)	Draws text on the control.
	GetItemHeight (see page 3779)	This is GetItemHeight, a member of class TRibbonRichContentItem.
	GetItemWidth (see page 3780)	This is GetItemWidth, a member of class TRibbonRichContentItem.
	GetMenuProps (see page 3780)	This is GetMenuProps, a member of class TRibbonRichContentItem.
	GetShowShortCut (see page 3759)	This is GetShowShortCut, a member of class TRibbonMenuItem.
	RibbonStyle (see page 3760)	This is RibbonStyle, a member of class TRibbonMenuItem.

Protected Properties

	Name	Description
	DrawBanner (see page 3760)	This is DrawBanner, a member of class TRibbonMenuItem.

Public Methods

	Name	Description
	CalcBounds (see page 3777)	Calculates the size of the menu item.
	Click (see page 3757)	Executes the control's function.
	Create (see page 3777)	Creates an instance of TCustomMenuItem.

63.1.38.2 TRibbonRichContentItem.Create Constructor

Creates an instance of TCustomMenuItem.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonRichContentItem(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomMenuItem. However, you should never attempt to instantiate a TCustomMenuItem. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

After calling the inherited constructor, Create initializes the control, including the following property settings:

Initializes the TCanvas object for its Canvas property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Create

Destroy

63.1.38.3 TRibbonRichContentItem Methods

The methods of the TRibbonRichContentItem class are listed here.

Protected Methods

	Name	Description
	CanCalculateHeight (see page 3778)	This is CanCalculateHeight, a member of class TRibbonRichContentItem.
V	DrawBackground (see page 3778)	Draws a background image on the control.
V	DrawGlyph (see page 3778)	Draws a symbol on the control.
V	DrawShadowedText (see page 3779)	This is DrawShadowedText, a member of class TRibbonRichContentItem.
V	DrawText (see page 3779)	Draws text on the control.
	GetItemHeight (see page 3779)	This is GetItemHeight, a member of class TRibbonRichContentItem.
	GetItemWidth (see page 3780)	This is GetItemWidth, a member of class TRibbonRichContentItem.
	GetMenuProps (see page 3780)	This is GetMenuProps, a member of class TRibbonRichContentItem.

Public Methods

	Name	Description
V	CalcBounds (see page 3777)	Calculates the size of the menu item.

63.1.38.3.1 TRibbonRichContentItem.CalcBounds Method

Calculates the size of the menu item.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as whether or not the menu item is a separator or not. CalcBounds computes the boundaries of the menu item considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.38.3.2 TRibbonRichContentItem.CanCalculateHeight Method

Pascal

```
function CanCalculateHeight: Boolean;
```

C++

```
__fastcall Boolean CanCalculateHeight();
```

Description

This is CanCalculateHeight, a member of class TRibbonRichContentItem.

63.1.38.3.3 TRibbonRichContentItem.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.38.3.4 TRibbonRichContentItem.DrawGlyph Method

Draws a symbol on the control.

Pascal

```
procedure DrawGlyph(const Location: TPoint); override;
```

C++

```
virtual __fastcall DrawGlyph(const TPoint Location);
```

Description

Use DrawGlyph to draw a symbol on the control at the specified location.

See Also

DrawBackground

DrawText

63.1.38.3.5 TRibbonRichContentItem.DrawShadowedText Method

Pascal

```
procedure DrawShadowedText(Rect: TRect; Flags: Cardinal; Text: string; TextColor: TColor;
ShadowColor: TColor); override;
```

C++

```
virtual __fastcall DrawShadowedText(TRect Rect, unsigned Flags, UnicodeString Text, TColor
TextColor, TColor ShadowColor);
```

Description

This is DrawShadowedText, a member of class TRibbonRichContentItem.

63.1.38.3.6 TRibbonRichContentItem.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var Rect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect Rect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

See Also

DrawBackground

DrawGlyph

63.1.38.3.7 TRibbonRichContentItem.GetItemHeight Method

Pascal

```
function GetItemHeight: Integer;
```

C++

```
__fastcall int GetItemHeight();
```

Description

This is GetItemHeight, a member of class TRibbonRichContentItem.

63.1.38.3.8 TRibbonRichContentItem.GetItemWidth Method

Pascal

```
function GetItemWidth: Integer;
```

C++

```
__fastcall int GetItemWidth();
```

Description

This is GetItemWidth, a member of class TRibbonRichContentItem.

63.1.38.3.9 TRibbonRichContentItem.GetMenuProps Method

Pascal

```
function GetMenuProps: TMenuProperties;
```

C++

```
__fastcall TMenuProperties GetMenuProps();
```

Description

This is GetMenuProps, a member of class TRibbonRichContentItem.

63.1.39 TRibbonRichContentMainMenuItem Class

Class Hierarchy

Pascal

```
TRibbonRichContentMainMenuItem = class(TRibbonRichContentItem);
```

C++

```
class TRibbonRichContentMainMenuItem : public TRibbonRichContentItem;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonRichContentMainMenuItem.

63.1.39.1 TRibbonRichContentMainMenuItem Members

The following tables list the members exposed by TRibbonRichContentMainMenuItem.

Protected Methods

	Name	Description
	CalcLayout (see page 3782)	This is CalcLayout, a member of class TRibbonRichContentMainMenuItem.
	CanCalculateHeight (see page 3778)	This is CanCalculateHeight, a member of class TRibbonRichContentItem.
	DrawBackground (see page 3758)	Draws a background image on the control.

	DrawBackground (see page 3778)	Draws a background image on the control.
	DrawGlyph (see page 3782)	Draws a symbol on the control.
	DrawSelectedBackground (see page 3758)	This is DrawSelectedBackground, a member of class TRibbonMenuItem.
	DrawSeparator (see page 3759)	This is DrawSeparator, a member of class TRibbonMenuItem.
	DrawShadowedText (see page 3759)	This is DrawShadowedText, a member of class TRibbonMenuItem.
	DrawShadowedText (see page 3779)	This is DrawShadowedText, a member of class TRibbonRichContentItem.
	DrawText (see page 3783)	Draws text on the control.
	GetImageSize (see page 3783)	This is GetImageSize, a member of class TRibbonRichContentMainMenuItem.
	GetItemHeight (see page 3779)	This is GetItemHeight, a member of class TRibbonRichContentItem.
	GetItemWidth (see page 3780)	This is GetItemWidth, a member of class TRibbonRichContentItem.
	GetMenuProps (see page 3780)	This is GetMenuProps, a member of class TRibbonRichContentItem.
	GetShowShortCut (see page 3783)	This is GetShowShortCut, a member of class TRibbonRichContentMainMenuItem.
	RibbonStyle (see page 3760)	This is RibbonStyle, a member of class TRibbonMenuItem.

Protected Properties

	Name	Description
	DrawBanner (see page 3760)	This is DrawBanner, a member of class TRibbonMenuItem.

Public Methods

	Name	Description
	CalcBounds (see page 3782)	Calculates the size of the menu item.
	Click (see page 3757)	Executes the control's function.
	Create (see page 3781)	Creates an instance of TCustomMenuItem.

63.1.39.2 TRibbonRichContentMainMenuItem.Create Constructor

Creates an instance of TCustomMenuItem.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonRichContentMainMenuItem(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomMenuItem. However, you should never attempt to instantiate a TCustomMenuItem. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

After calling the inherited constructor, Create initializes the control, including the following property settings:

Initializes the TCanvas object for its Canvas property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Create

Destroy

63.1.39.3 TRibbonRichContentMainMenuItem Methods

The methods of the TRibbonRichContentMainMenuItem class are listed here.

Protected Methods

	Name	Description
	CalcLayout (see page 3782)	This is CalcLayout, a member of class TRibbonRichContentMainMenuItem.
	DrawGlyph (see page 3782)	Draws a symbol on the control.
	DrawText (see page 3783)	Draws text on the control.
	GetImageSize (see page 3783)	This is GetImageSize, a member of class TRibbonRichContentMainMenuItem.
	GetShowShortCut (see page 3783)	This is GetShowShortCut, a member of class TRibbonRichContentMainMenuItem.

Public Methods

	Name	Description
	CalcBounds (see page 3782)	Calculates the size of the menu item.

63.1.39.3.1 TRibbonRichContentMainMenuItem.CalcBounds Method

Calculates the size of the menu item.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as whether or not the menu item is a separator or not. CalcBounds computes the boundaries of the menu item considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.39.3.2 TRibbonRichContentMainMenuItem.CalcLayout Method

Pascal

```
procedure CalcLayout; override;
```

C++

```
virtual __fastcall CalcLayout();
```

Description

This is CalcLayout, a member of class TRibbonRichContentMainMenuItem.

63.1.39.3.3 TRibbonRichContentMainMenuItem.DrawGlyph Method

Draws a symbol on the control.

Pascal

```
procedure DrawGlyph(const Location: TPoint); override;
```

C++

```
virtual __fastcall DrawGlyph(const TPoint Location);
```

Description

Use DrawGlyph to draw a symbol on the control at the specified location.

See Also

DrawBackground

DrawText

63.1.39.3.4 TRibbonRichContentMainMenuItem.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var ARect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect ARect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

See Also

DrawBackground

DrawGlyph

63.1.39.3.5 TRibbonRichContentMainMenuItem.GetImageSize Method

Pascal

```
function GetImageSize: TPoint; override;
```

C++

```
virtual __fastcall TPoint GetImageSize();
```

Description

This is GetImageSize, a member of class TRibbonRichContentMainMenuItem.

63.1.39.3.6 TRibbonRichContentMainMenuItem.GetShowShortCut Method

Pascal

```
function GetShowShortCut: Boolean; override;
```

C++

```
virtual __fastcall Boolean GetShowShortCut();
```

Description

This is GetShowShortCut, a member of class TRibbonRichContentMainMenuItem.

63.1.40 TRibbonSeparator Class

Class Hierarchy

Pascal

```
TRibbonSeparator = class(TCustomActionControl);
```

C++

```
class TRibbonSeparator : public TCustomActionControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonSeparator.

63.1.40.1 TRibbonSeparator Members

The following tables list the members exposed by TRibbonSeparator.

Protected Methods

	Name	Description
	DrawBackground (see page 3785)	Draws a background image on the control.
	DrawGlyph (see page 3786)	Draws a symbol on the control.
	DrawShadowedText (see page 3786)	This is DrawShadowedText, a member of class TRibbonSeparator.
	DrawText (see page 3786)	Draws text on the control.
	RibbonStyle (see page 3787)	This is RibbonStyle, a member of class TRibbonSeparator.

Public Methods

	Name	Description
	CalcBounds (see page 3785)	Calculates the size of the control.
	Create (see page 3784)	Creates an instance of TCustomActionControl.

63.1.40.2 TRibbonSeparator.Create Constructor

Creates an instance of TCustomActionControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonSeparator(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomActionControl. However, you should never attempt to instantiate a TCustomActionControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

AOwner is the component that is responsible for freeing the TCustomActionControl instance. It becomes the value of the Owner property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Owner

Create

Destroy

63.1.40.3 TRibbonSeparator Methods

The methods of the TRibbonSeparator class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3785)	Draws a background image on the control.
	DrawGlyph (see page 3786)	Draws a symbol on the control.
	DrawShadowedText (see page 3786)	This is DrawShadowedText, a member of class TRibbonSeparator.
	DrawText (see page 3786)	Draws text on the control.
	RibbonStyle (see page 3787)	This is RibbonStyle, a member of class TRibbonSeparator.

Public Methods

	Name	Description
	CalcBounds (see page 3785)	Calculates the size of the control.

63.1.40.3.1 TRibbonSeparator.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as the presence of an icon, caption, and shortcuts. CalcBounds computes the boundaries of the control considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.40.3.2 TRibbonSeparator.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.40.3.3 TRibbonSeparator.DrawGlyph Method

Draws a symbol on the control.

Pascal

```
procedure DrawGlyph(const Location: TPoint); override;
```

C++

```
virtual __fastcall DrawGlyph(const TPoint Location);
```

Description

Use DrawGlyph to draw a symbol on the control at the specified location.

See Also

DrawBackground

DrawText

63.1.40.3.4 TRibbonSeparator.DrawShadowedText Method

Pascal

```
procedure DrawShadowedText(Rect: TRect; Flags: Cardinal; Text: string; TextColor: TColor;
ShadowColor: TColor); override;
```

C++

```
virtual __fastcall DrawShadowedText(TRect Rect, unsigned Flags, UnicodeString Text, TColor
TextColor, TColor ShadowColor);
```

Description

This is DrawShadowedText, a member of class TRibbonSeparator.

63.1.40.3.5 TRibbonSeparator.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var ARect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect ARect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

See Also

DrawBackground

DrawGlyph

63.1.40.3.6 TRibbonSeparator.RibbonStyle Method

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

This is RibbonStyle, a member of class TRibbonSeparator.

63.1.41 TRibbonSmallButtonControl Class

Class Hierarchy

Pascal

```
TRibbonSmallButtonControl = class(TRibbonBaseButtonControl);
```

C++

```
class TRibbonSmallButtonControl : public TRibbonBaseButtonControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonSmallButtonControl.

63.1.41.1 TRibbonSmallButtonControl Members

The following tables list the members exposed by TRibbonSmallButtonControl.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	DrawBackground (see page 3621)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.
	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.

	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawText (see page 3622)	Draws text on the control.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3788)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3620)	Executes the control's function.
	Create (see page 3617)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
	KeyTipClick (see page 3624)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.41.2 TRibbonSmallButtonControl Methods

The methods of the TRibbonSmallButtonControl class are listed here.

Public Methods

	Name	Description
	CalcBounds (see page 3788)	Calculates the size of the control.

63.1.41.2.1 TRibbonSmallButtonControl.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

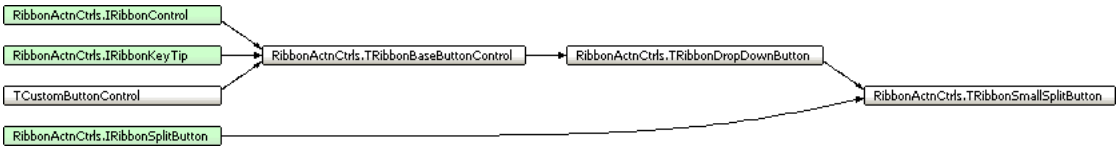
This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.42 TRibbonSmallSplitButton Class

Class Hierarchy



Pascal

```
TRibbonSmallSplitButton = class(TRibbonDropDownButton, IRibbonSplitButton);
```

C++

```
class TRibbonSmallSplitButton : public TRibbonDropDownButton, public IRibbonSplitButton;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonSmallSplitButton.

63.1.42.1 TRibbonSmallSplitButton Members

The following tables list the members exposed by TRibbonSmallSplitButton.

Protected Fields

	Name	Description
	FCursorInSplit (see page 3791)	This is FCursorInSplit, a member of class TRibbonSmallSplitButton.
	FSplitRect (see page 3791)	This is FSplitRect, a member of class TRibbonSmallSplitButton.

Protected Methods

	Name	Description
	AllowHotKeys (see page 3619)	This is AllowHotKeys, a member of class TRibbonBaseButtonControl.
	CalcLayout (see page 3792)	This is CalcLayout, a member of class TRibbonSmallSplitButton.
	CMMouseLeave (see page 3793)	This is CMMouseLeave, a member of class TRibbonSmallSplitButton.
	DoDropDown (see page 3718)	This is DoDropDown, a member of class TRibbonDropDownButton.
	DrawArrow (see page 3718)	This is DrawArrow, a member of class TRibbonDropDownButton.
	DrawBackground (see page 3793)	Draws a background image on the control.
	DrawGlyph (see page 3621)	Draws a symbol on the control.

	DrawSeparator (see page 3621)	This is DrawSeparator, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3622)	This is DrawShadowedText, a member of class TRibbonBaseButtonControl.
	DrawShadowedText (see page 3718)	This is DrawShadowedText, a member of class TRibbonDropDownButton.
	DrawText (see page 3622)	Draws text on the control.
	DrawText (see page 3719)	Draws text on the control.
	GetArrowPosition (see page 3794)	This is GetArrowPosition, a member of class TRibbonSmallSplitButton.
	GetImageSize (see page 3622)	This is GetImageSize, a member of class TRibbonBaseButtonControl.
	GetPopupClass (see page 3719)	This is GetPopupClass, a member of class TRibbonDropDownButton.
	InSplit (see page 3794)	This is InSplit, a member of class TRibbonSmallSplitButton.
	IsButton (see page 3624)	This is IsButton, a member of class TRibbonBaseButtonControl.
	IsRibbonAvailable (see page 3624)	This is IsRibbonAvailable, a member of class TRibbonBaseButtonControl.
	MouseDown (see page 3794)	OnMouseDown event dispatcher.
	MouseMove (see page 3795)	Respond to mouse moving over control area..
	RibbonStyle (see page 3624)	This is RibbonStyle, a member of class TRibbonBaseButtonControl.

Protected Properties

	Name	Description
	DrawDown (see page 3626)	This is DrawDown, a member of class TRibbonBaseButtonControl.

Public Methods

	Name	Description
	CalcBounds (see page 3792)	Calculates the size of the control.
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	CanAddToQuickAccessToolbar (see page 3620)	This is CanAddToQuickAccessToolbar, a member of class TRibbonBaseButtonControl.
	Click (see page 3793)	Executes the control's function.
	Create (see page 3790)	Creates an instance of TCustomButtonControl.
	Destroy (see page 3618)	Destroys an instance of TCustomActionControl.
	DragDrop (see page 3620)	Performs actions and generates an OnDragDrop event.
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	GetKeyTip (see page 3622)	Returns current **KeyTip** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HandleKeyTip (see page 3623)	Indicates if command's **KeyTip** matches currently input KeyTip.
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	HideKeyTip (see page 3623)	Hide **KeyTip** for command.
	KeyTipClick (see page 3794)	Handle clicking a **KeyTip**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	SetKeyTip (see page 3625)	Sets **KeyTip** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.
	ShowKeyTip (see page 3625)	Shows **KeyTip** for command in specified position.
	UpdateSplitBounds (see page 3795)	This is UpdateSplitBounds, a member of class TRibbonSmallSplitButton.

Public Properties

	Name	Description
	CursorInSplit (see page 3796)	This is CursorInSplit, a member of class TRibbonSmallSplitButton.
	KeyTip (see page 3922)	Current **KeyTip** for command

63.1.42.2 TRibbonSmallSplitButton.Create Constructor

Creates an instance of TCustomButtonControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonSmallSplitButton(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomButtonControl. However, you should never attempt to instantiate a TCustomButtonControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

AOwner is the component that is responsible for freeing the button control. It becomes the value of the Owner property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Owner

Create

Destroy

63.1.42.3 TRibbonSmallSplitButton Fields

The fields of the TRibbonSmallSplitButton class are listed here.

Protected Fields

	Name	Description
	FCursorInSplit (see page 3791)	This is FCursorInSplit, a member of class TRibbonSmallSplitButton.
	FSplitRect (see page 3791)	This is FSplitRect, a member of class TRibbonSmallSplitButton.

63.1.42.3.1 TRibbonSmallSplitButton.FCursorInSplit Field

Pascal

```
FCursorInSplit: Boolean;
```

C++

```
Boolean FCursorInSplit;
```

Description

This is FCursorInSplit, a member of class TRibbonSmallSplitButton.

63.1.42.3.2 TRibbonSmallSplitButton.FSplitRect Field

Pascal

```
FSplitRect: TRect;
```

C++

```
TRect FSplitRect;
```

Description

This is FSplitRect, a member of class TRibbonSmallSplitButton.

63.1.42.4 TRibbonSmallSplitButton Methods

The methods of the TRibbonSmallSplitButton class are listed here.

Protected Methods

	Name	Description
	CalcLayout (see page 3792)	This is CalcLayout, a member of class TRibbonSmallSplitButton.
	CMMouseLeave (see page 3793)	This is CMMouseLeave, a member of class TRibbonSmallSplitButton.
	DrawBackground (see page 3793)	Draws a background image on the control.
	GetArrowPosition (see page 3794)	This is GetArrowPosition, a member of class TRibbonSmallSplitButton.
	InSplit (see page 3794)	This is InSplit, a member of class TRibbonSmallSplitButton.
	MouseDown (see page 3794)	OnMouseDown event dispatcher.
	MouseMove (see page 3795)	Respond to mouse moving over control area..

Public Methods

	Name	Description
	CalcBounds (see page 3792)	Calculates the size of the control.
	Click (see page 3793)	Executes the control's function.
	KeyTipClick (see page 3794)	Handle clicking a **KeyTip**.
	UpdateSplitBounds (see page 3795)	This is UpdateSplitBounds, a member of class TRibbonSmallSplitButton.

63.1.42.4.1 TRibbonSmallSplitButton.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

This method is largely the same as CalcBounds (TCustomActionControl), with one minor exception. The width of the control is set to be one pixel wider. For more information, see the documentation for the inherited method.

See Also

CalcBounds

63.1.42.4.2 TRibbonSmallSplitButton.CalcLayout Method

Pascal

```
procedure CalcLayout; override;
```

C++

```
virtual __fastcall CalcLayout();
```

Description

This is CalcLayout, a member of class TRibbonSmallSplitButton.

63.1.42.4.3 TRibbonSmallSplitButton.Click Method

Executes the control's function.

Pascal

```
procedure Click; override;
```

C++

```
virtual __fastcall Click();
```

Description

This override of the method inherited from TCustomActionControl is almost identical to the inherited method. The only difference is that if the button control is being manipulated at design time, extra actions are taken to ensure that the control is properly selected.

See Also

Click

Click

UsageCount

63.1.42.4.4 TRibbonSmallSplitButton.CMMouseLeave Method

Pascal

```
procedure CMMouseLeave(var Message: TMessage); override;
```

C++

```
virtual __fastcall CMMouseLeave(TMessage Message);
```

Description

This is CMMouseLeave, a member of class TRibbonSmallSplitButton.

63.1.42.4.5 TRibbonSmallSplitButton.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.42.4.6 TRibbonSmallSplitButton.GetArrowPosition Method

Pascal

```
function GetArrowPosition(const ARect: TRect): TPoint; override;
```

C++

```
virtual __fastcall TPoint GetArrowPosition(const TRect ARect);
```

Description

This is GetArrowPosition, a member of class TRibbonSmallSplitButton.

63.1.42.4.7 TRibbonSmallSplitButton.InSplit Method

Pascal

```
function InSplit(X: Integer; Y: Integer): Boolean;
```

C++

```
__fastcall Boolean InSplit(int X, int Y);
```

Description

This is InSplit, a member of class TRibbonSmallSplitButton.

63.1.42.4.8 TRibbonSmallSplitButton.KeyTipClick Method

Handle clicking a **KeyTip**.

Pascal

```
procedure KeyTipClick; override;
```

C++

```
virtual __fastcall KeyTipClick();
```

Description

KeyTipClick Handle clicking a **KeyTip**. Some requires need to be handled differently when clicked by a **KeyTip**.

63.1.42.4.9 TRibbonSmallSplitButton.MouseDown Method

OnMouseDown event dispatcher.

Pascal

```
procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer);
override;
```

C++

```
virtual __fastcall MouseDown(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user presses the mouse button down while the cursor's hotspot is over the control.

The Button parameter determines which mouse button the user pressed. Shift indicates which shift keys (Shift, Ctrl, or Alt) were down when the user pressed the mouse button. X and Y are the pixel coordinates of the mouse pointer within the client area of the control.

A control calls MouseDown in response to any of the Windows mouse-down messages (WM_LBUTTONDOWN, WM_MBUTTONDOWN, WM_RBUTTONDOWN), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was pressed down: left, right, or middle.

See Also

MouseMove

MouseUp

OnMouseDown

63.1.42.4.10 TRibbonSmallSplitButton.MouseMove Method

Respond to mouse moving over control area..

Pascal

```
procedure MouseMove(Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseMove(TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseMove method to provide other responses in addition to calling the OnMouseMove event handler when the user moves the mouse.

A control calls MouseMove in response to any of the Windows mouse-move messages (WM_MOUSEMOVE), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively.

As the mouse cursor moves across a control, this method is called repeatedly. Each time it is called, it is with the new coordinates that reflect the continuous path of the mouse cursor across the screen real estate covered by the control's visual representation.

See Also

MouseDown

MouseUp

OnMouseMove

63.1.42.4.11 TRibbonSmallSplitButton.UpdateSplitBounds Method

Pascal

```
procedure UpdateSplitBounds; virtual;
```

C++

```
virtual __fastcall UpdateSplitBounds();
```

Description

This is UpdateSplitBounds, a member of class TRibbonSmallSplitButton.

63.1.42.5 TRibbonSmallSplitButton Properties

The properties of the TRibbonSmallSplitButton class are listed here.

Public Properties

	Name	Description
	CursorInSplit (see page 3796)	This is CursorInSplit, a member of class TRibbonSmallSplitButton.

63.1.42.5.1 TRibbonSmallSplitButton.CursorInSplit Property

Pascal

```
property CursorInSplit: Boolean;
```

C++

```
__property Boolean CursorInSplit;
```

Description

This is CursorInSplit, a member of class TRibbonSmallSplitButton.

63.1.43 TRibbonSpinButton Class

Class Hierarchy

Pascal

```
TRibbonSpinButton = class(TWinControl);
```

C++

```
class TRibbonSpinButton : public TWinControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonSpinButton.

63.1.43.1 TRibbonSpinButton Members

The following tables list the members exposed by TRibbonSpinButton.

Protected Methods

	Name	Description
	Control (see page 3798)	This is Control, a member of class TRibbonSpinButton.
	KeyDown (see page 3798)	Respond to key press events.
	Loaded (see page 3799)	Initializes the control after it is loaded from a stream.
	Notification (see page 3799)	Responds to notifications that components are being created or destroyed.

Public Methods

	Name	Description
	Create (see page 3797)	Creates an instance of TWinControl.
	SetBounds (see page 3800)	Sets the windowed control's boundary properties all at once.

Published Events

	Name	Description
	OnDownClick (see page 3820)	This is OnDownClick, a member of class TRibbonSpinButton.
	OnDragDrop (see page 3820)	Occurs when the user drops an object being dragged.
	OnDragOver (see page 3823)	Occurs when the user drags an object over a control.
	OnEndDock (see page 3827)	Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.
	OnEndDrag (see page 3827)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.
	OnEnter (see page 3828)	Occurs when a control receives the input focus.
	OnExit (see page 3831)	Occurs when the input focus shifts away from one control to another.
	OnStartDock (see page 3835)	Occurs when the user begins to drag a control with a DragKind of dkDock.
	OnStartDrag (see page 3835)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.
	OnUpClick (see page 3836)	This is OnUpClick, a member of class TRibbonSpinButton.

Published Properties

	Name	Description
	Align (see page 3801)	Determines how the control aligns within its container (parent control).
	Anchors (see page 3804)	Specifies how the control is anchored to its parent.
	Constraints (see page 3805)	Specifies the size constraints for the control.
	Ctl3D (see page 3806)	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
	DownGlyph (see page 3806)	This is DownGlyph, a member of class TRibbonSpinButton.
	DownNumGlyphs (see page 3806)	This is DownNumGlyphs, a member of class TRibbonSpinButton.
	DragCursor (see page 3807)	Indicates the image used to represent the mouse pointer when the control is being dragged.
	DragKind (see page 3807)	Specifies whether the control is being dragged normally or for docking.
	DragMode (see page 3807)	Determines how the control initiates drag-and-drop or drag-and-dock operations.
	Enabled (see page 3809)	Controls whether the control responds to mouse, keyboard, and timer events.
	FocusControl (see page 3809)	This is FocusControl, a member of class TRibbonSpinButton.
	ParentCtl3D (see page 3810)	Determines where a component looks to determine if it should appear three dimensional.
	ParentShowHint (see page 3810)	Determines where a control looks to find out if its Help Hint should be shown.
	PopupMenu (see page 3810)	Identifies the pop-up menu associated with the control.
	ShowHint (see page 3815)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	TabOrder (see page 3817)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 3818)	Determines if the user can tab to a control.
	UpGlyph (see page 3818)	This is UpGlyph, a member of class TRibbonSpinButton.
	UpNumGlyphs (see page 3818)	This is UpNumGlyphs, a member of class TRibbonSpinButton.
	Visible (see page 3818)	Determines whether the component appears onscreen.

63.1.43.2 TRibbonSpinButton.Create Constructor

Creates an instance of TWinControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonSpinButton(TComponent * AOwner);
```

Description

Call Create to construct and initialize a new control and insert the newly-constructed control into its owner, as specified by the

AOwner parameter. Create inserts the control in the owner by calling the owner's InsertComponent method.

Most controls override Create to initialize their unique properties. Objects that override the Create method must always call the inherited Create method first, and then proceed with the component-specific initialization. Specify the override directive when overriding the Create method.

If a component's Create method allocates resources or memory, override the Destroy method to free those resources.

See Also

Brush

TabOrder

CreateParented

InsertComponent

Create

Owner

63.1.43.3 TRibbonSpinButton Methods

The methods of the TRibbonSpinButton class are listed here.

Protected Methods

	Name	Description
	Control (see page 3798)	This is Control, a member of class TRibbonSpinButton.
V	KeyDown (see page 3798)	Respond to key press events.
V	Loaded (see page 3799)	Initializes the control after it is loaded from a stream.
V	Notification (see page 3799)	Responds to notifications that components are being created or destroyed.

Public Methods

	Name	Description
V	SetBounds (see page 3800)	Sets the windowed control's boundary properties all at once.

63.1.43.3.1 TRibbonSpinButton.Control Method

Pascal

```
function Control: TRibbonSpinEdit;
```

C++

```
__fastcall TRibbonSpinEdit Control();
```

Description

This is Control, a member of class TRibbonSpinButton.

63.1.43.3.2 TRibbonSpinButton.KeyDown Method

Respond to key press events.

Pascal

```
procedure KeyDown(var Key: Word; Shift: TShiftState); override;
```

C++

```
virtual __fastcall KeyDown(Word Key, TShiftState Shift);
```

Description

When a windowed control receives a key-down message (WM_KEYDOWN) from Windows, its message handler calls the DoKeyDown method. If DoKeyDown determines that the control should, in fact, process the character, it calls KeyDown, passing the key code and shift-key state in the Key and Shift parameters, respectively.

KeyDown calls any event handler attached to the OnKeyDown event. Override KeyDown to provide other responses in addition to the event-handler call.

The Key parameter is the key on the keyboard. For non-alphanumeric keys, you must use WinAPI virtual key codes to determine the key pressed. For more information, search for virtual key codes in the Win32 Developer's Reference (WIN32.HLP).

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

Either KeyDown or the OnKeyDown event handler it calls can suppress further processing of a key by setting the Key parameter to zero.

See Also

KeyUp

OnKeyDown

DoKeyDown

DoKeyPress

DoKeyUp

63.1.43.3.3 TRibbonSpinButton.Loaded Method

Initializes the control after it is loaded from a stream.

Pascal

```
procedure Loaded; override;
```

C++

```
virtual __fastcall Loaded();
```

Description

Do not call Loaded. The VCL streaming system calls Loaded automatically after the control's form is loaded into memory so that the control can complete any initializations that depend on other objects in the form.

Loaded overrides the inherited method in order to initialize the control from its associated Action. To change the properties the control copies from its action, override the ActionChange method.

See Also

Loaded

Action

ActionChange

63.1.43.3.4 TRibbonSpinButton.Notification Method

Responds to notifications that components are being created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification allows TControl to update its PopupMenu property when the pop-up menu it refers to is destroyed.

Override the Notification method to respond to notifications that other components are about to be destroyed or have just been created. Use the Notification method to update controls that rely on other objects. Remove references to objects that are going away so that the control does not refer to objects that have been destroyed. For example, data-aware objects override the Notification method to update themselves when their data source is removed. Some objects respond to notifications that other objects of a specific type are being created. For example, the session component sets the Session property of new data-aware controls to itself when the AutoSessionName property is true.

By default, components pass along the notification to their owned components, if any.

See Also

PopupMenu

Notification

63.1.43.3.5 TRibbonSpinButton.SetBounds Method

Sets the windowed control's boundary properties all at once.

Pascal

```
procedure SetBounds(ALeft: Integer; ATop: Integer; AWidth: Integer; AHeight: Integer);
override;
```

C++

```
virtual __fastcall SetBounds(int ALeft, int ATop, int AWidth, int AHeight);
```

Description

Use SetBounds to change all of the control's boundary properties at once. The same effect can be achieved by setting the Left, Top, Width, and Height properties separately. By setting all four properties at once, SetBounds ensures that the control will not repaint between changes

Specify the values for the Left, Top, Width, and Height properties as the value of the ALeft, ATop, AWidth, and AHeight parameters, respectively.

See Also

Height

Left

Top

Width

63.1.43.4 TRibbonSpinButton Properties

The properties of the TRibbonSpinButton class are listed here.

Published Properties

	Name	Description
	Align (see page 3801)	Determines how the control aligns within its container (parent control).
	Anchors (see page 3804)	Specifies how the control is anchored to its parent.
	Constraints (see page 3805)	Specifies the size constraints for the control.
	Ctl3D (see page 3806)	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
	DownGlyph (see page 3806)	This is DownGlyph, a member of class TRibbonSpinButton.
	DownNumGlyphs (see page 3806)	This is DownNumGlyphs, a member of class TRibbonSpinButton.
	DragCursor (see page 3807)	Indicates the image used to represent the mouse pointer when the control is being dragged.
	DragKind (see page 3807)	Specifies whether the control is being dragged normally or for docking.
	DragMode (see page 3807)	Determines how the control initiates drag-and-drop or drag-and-dock operations.
	Enabled (see page 3809)	Controls whether the control responds to mouse, keyboard, and timer events.
	FocusControl (see page 3809)	This is FocusControl, a member of class TRibbonSpinButton.
	ParentCtl3D (see page 3810)	Determines where a component looks to determine if it should appear three dimensional.
	ParentShowHint (see page 3810)	Determines where a control looks to find out if its Help Hint should be shown.
	PopupMenu (see page 3810)	Identifies the pop-up menu associated with the control.
	ShowHint (see page 3815)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	TabOrder (see page 3817)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 3818)	Determines if the user can tab to a control.
	UpGlyph (see page 3818)	This is UpGlyph, a member of class TRibbonSpinButton.
	UpNumGlyphs (see page 3818)	This is UpNumGlyphs, a member of class TRibbonSpinButton.
	Visible (see page 3818)	Determines whether the component appears onscreen.

63.1.43.4.1 TRibbonSpinButton.Align Property

Determines how the control aligns within its container (parent control).

Pascal

```
property Align;
```

C++

```
__property Align;
```

Description

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes. When the parent is resized, an aligned control also resizes so that it continues to span the top, bottom, left, or right edge of the parent.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel.

Tip: If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

Note: To cause a control to maintain a specified relationship with an edge of its parent, but not necessarily lie along one edge of

the parent, use the Anchors property instead.

C++ Examples:

```
/*
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
*/

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  TSplitter* ps = new TSplitter(Form1); // These objects will all be cleaned up by the owner
(Form1).
  TFileListBox* pflb = new TFileListBox(Form1); // The parent (Form1) will clean up these
objects.
  TDirectoryListBox* pdlb = new TDirectoryListBox(Form1);

  // Line up the directory list box on the left of the form.
  pdlb->Parent = Form1;
  pdlb->Align = alLeft;
  pdlb->Width = Form1->ClientWidth/3;

  // Now use the splitter to divide the directory pane from the file pane.
  ps->Parent = Form1;
  ps->Left = pdlb->Left + pdlb->Width + 1; // Move to right of directory list.
  ps->Align = pdlb->Align; // Give same alignment as directory list.

  // Each pane must be at least one quarter of the form's width.
  ps->MinSize = Form1->ClientWidth/4;

  // Finally, create the last pane - a file list box.
  pflb->Parent = Form1;
  pflb->Align = alClient;
  pdlb->FileList = pflb; // Link the file list box to the directory list box.
}
/*
Dynamically create a TProgressBar control and align it to
the bottom of the form.
*/

#include <ComCtrls.hpp>
#include <Controls.hpp>

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TProgressBar *ProgressBar = new TProgressBar(this);  // The owner will clean this up.
  ProgressBar->Parent = this;
  ProgressBar->Align = alBottom;
  TButton *Button1 = new TButton(this);  // The owner will clean this up.
  Button1->Parent = this;
}
/*
This example dynamically creates a Page Control, then a
series of Tab Sheets on the Page Control.
*/
#include <Comctrls.hpp>
TPageControl* ppc;
const int MAXTABS = 3;
TTabSheet* pts[MAXTABS];
```

```
const char * ppcTabTitles[] = { "ShortString", "Orders", "Items", "Parts" };
int iTabTitles = sizeof(ppcTabTitles)/sizeof(ppcTabTitles[0]);
const TColor colorPalette[12] = {
  clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia,
  clTeal, clNavy, clMaroon, clLime, clOlive, clPurple};

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    ppc = new TPageControl(this); // The owner (this) will clean this up.
    ppc->Parent = this;
    ppc->Align = alClient;
    for (int i=0;i<iTabTitles;i++)
  {
        pts[i] = new TTabSheet(this); // The owner (this) will clean this up.
        pts[i]->PageControl = ppc;
        pts[i]->Name = AnsiString("pts") + ppcTabTitles[i];
        pts[i]->Caption = ppcTabTitles[i];
        pts[i]->Brush->Color = colorPalette[i];
  }
}
```

Delphi Examples:

```
{
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  Split: TSplitter;
  Files: TFileListBox;
  Dirs: TDirectoryListBox;
begin
  { first add a directory list box to the form }
  Dirs := TDirectoryListBox.Create(Form1);
  { line it up on the left of the form }
  Dirs.Parent := Form1;
  Dirs.Align := alLeft;
  Dirs.Width := Form1.ClientWidth div 3;
{ now add the splitter to divide the directory pane from the file pane }
  Split := TSplitter.Create(Form1);
  Split.Parent := Form1;
  { make sure the splitter is to the right of the directory list box! }
  Split.Left := Dirs.Left + Dirs.Width + 1;
  Split.Align := Dirs.Align; { give it the same alignment as the directory }
  { each pane must be at least one quarter of the form's width }
  Split.MinSize := Form1.ClientWidth div 4;
  { Finally, create the last pane – a file list box }
  Files := TFileListBox.Create(Form1);
  Files.Parent := Form1;
  Files.Align := alClient;
  Dirs.FileList := Files; { link the file list box to the directory list box }
end;
{
Dynamically create a TProgressBar control and align it to
the bottom of the form.
}

var
  I: Integer;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
  ProgressBar: TProgressBar;
begin
  ProgressBar := TProgressBar.Create(Self);
  with ProgressBar do
  begin
    Parent := Self;
    Align := alBottom;
  end;
end;
{
This example dynamically creates a Page Control, then a
series of Tab Sheets on the Page Control.
}
var
  PageControl1: TPageControl;
  pts: array[0..3] of TTabSheet;
const
  TabTitles: array[0..3] of ShortString = ('Customer', 'Orders', 'Items', 'Parts' );

procedure TForm1.FormCreate(Sender: TObject);
const
  colorpalette: Array[0..11] of TColor = (
    clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia, clTeal,
    clNavy, clMaroon, clLime, clOlive, clPurple);
var
  i: Integer;
begin
  PageControl1 := TPageControl.Create(Self);
  PageControl1.Parent := Self;
  PageControl1.Align := alClient;
  for i := Low(TabTitles) to High(TabTitles) do
  begin
    pts[i]:= TTabSheet.Create(PageControl1);
    with pts[i] do
    begin
      PageControl := PageControl1;
      Name := 'ts' + TabTitles[i];
      Caption := TabTitles[i];
      Brush.Color := colorPalette[i];
   end;
  end;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  i: Integer;
begin
  for i := Low(TabTitles) to High(TabTitles) do
  begin
    pts[i].Free;
  end;
  PageControl1.Free;
end;
```

See Also

Parent

Anchors

63.1.43.4.2 TRibbonSpinButton.Anchors Property

Specifies how the control is anchored to its parent.

Pascal

```
property Anchors;
```

C++

```
__property Anchors;
```

Description

Use Anchors to ensure that a control maintains its current position relative to an edge of its parent, even if the parent is resized. When its parent is resized, the control holds its position relative to the edges to which it is anchored.

If a control is anchored to opposite edges of its parent, the control stretches when its parent is resized. For example, if a control has its Anchors property set to [akLeft, akRight], the control stretches when the width of its parent changes.

Anchors is enforced only when the parent is resized. Thus, for example, if a control is anchored to opposite edges of a form at design time and the form is created in a maximized state, the control is not stretched because the form is not resized after the control is created.

Note: If a control should maintain contact with three edges of its parent (hugging one side of the parent and stretching the length of that side), use the Align property instead. Unlike Anchors, Align allows controls to adjust to changes in the size of other aligned sibling controls as well as changes to the parent's size.

See Also

Parent

Align

63.1.43.4.3 TRibbonSpinButton.Constraints Property

Specifies the size constraints for the control.

Pascal

```
property Constraints;
```

C++

```
__property Constraints;
```

Description

Use Constraints to specify the minimum and maximum width and height of the control. When Constraints contains maximum or minimum values, the control cannot be resized to violate those constraints.

Note: Constraints apply to the height of the form. However, the height of the form depends on how large Windows makes the title bar. For example, Windows XP uses much larger title bars than other windowing systems. To work around this, note the ClientHeight when you design your form and set the constraints in the FormCreate event. In the following example, ClientHeight is represented by x: `Constraints.MinHeight := x + Height - ClientHeight`.

Warning: Do not set up constraints that conflict with the value of the Align or Anchors property. When these properties conflict, the response of the control to resize attempts is not well-defined.

See Also

AutoSize

Align

Anchors

OnConstrainedResize

OnCanResize

63.1.43.4.4 TRibbonSpinButton.Ctl3D Property

Determines whether a control has a three-dimensional (3-D) or two-dimensional look.

Pascal

```
property Ctl3D;
```

C++

```
__property Ctl3D;
```

Description

Ctl3D is provided for backward compatibility. It is not used by 32-bit versions of Windows or NT4.0 and later, with the exception of Windows XP Home, where it is used on occasion.

On earlier platforms, Ctl3D controlled whether the control had a flat or beveled appearance.

See Also

ParentCtl3D

BevelKind

BevelInner

BevelOuter

BorderWidth

63.1.43.4.5 TRibbonSpinButton.DownGlyph Property

Pascal

```
property DownGlyph: TBitmap;
```

C++

```
__property TBitmap DownGlyph;
```

Description

This is DownGlyph, a member of class TRibbonSpinButton.

63.1.43.4.6 TRibbonSpinButton.DownNumGlyphs Property

Pascal

```
property DownNumGlyphs: TNumGlyphs;
```

C++

```
__property TNumGlyphs DownNumGlyphs;
```

Description

This is DownNumGlyphs, a member of class TRibbonSpinButton.

63.1.43.4.7 TRibbonSpinButton.DragCursor Property

Indicates the image used to represent the mouse pointer when the control is being dragged.

Pascal

```
property DragCursor;
```

C++

```
__property DragCursor;
```

Description

Use the DragCursor property to change the cursor image presented when the control is being dragged.

Note: To make a custom cursor available for the DragCursor property, see the Cursor property.

See Also

Cursor

DragKind

DragMode

63.1.43.4.8 TRibbonSpinButton.DragKind Property

Specifies whether the control is being dragged normally or for docking.

Pascal

```
property DragKind;
```

C++

```
__property DragKind;
```

Description

Use DragKind to get or set whether the control participates in drag-and-drop operations, or drag-and-dock operations.

See Also

DragCursor

DragMode

63.1.43.4.9 TRibbonSpinButton.DragMode Property

Determines how the control initiates drag-and-drop or drag-and-dock operations.

Pascal

```
property DragMode;
```

C++

```
__property DragMode;
```

Description

Use DragMode to control when the user can drag the control. Disable the drag-and-drop or drag-and-dock capability at runtime by setting the DragMode property value to dmManual. Enable automatic dragging by setting DragMode to dmAutomatic.

C++ Examples:

```
/*
The following code uses GetNodeAt to add a dragged node as a
child of the node under the mouse when it is dropped.  This
example requires a populated TreeView.  Also, the TTreeView
DragMode property set to dmAutomatic, and the TTreeView
OnDragOver event handler must be implemented to accept the
drop.
*/
void __fastcall TForm1::TreeView1DragDrop(TObject *Sender, TObject *Source,
      int X, int Y)
{
  if (Source != dynamic_cast<TObject *>(TreeView1) || TreeView1->Selected == NULL)
    return;
  THitTests HT = TreeView1->GetHitTestInfoAt(X, Y);
  TNodeAttachMode AttachMode;
  TTreeNode *Item = TreeView1->GetNodeAt(X, Y);
  if (HT.Contains(htOnItem) || HT.Contains(htOnIcon))
    AttachMode = naAddChild;
  else if (HT.Contains(htNowhere))
    AttachMode = naAdd;
  else if (HT.Contains(htOnIndent))
    AttachMode = naInsert;
  else
    return;
  TreeView1->Selected->MoveTo(Item, AttachMode);
}

void __fastcall TForm1::TreeView1DragOver(TObject *Sender, TObject *Source,
      int X, int Y, TDragState State, bool &Accept)
{
  Accept = (Source == dynamic_cast<TObject *>(TreeView1));
}
```

Delphi Examples:

```
{
The following code uses GetNodeAt to add a dragged node as a
child of the node under the mouse when it is dropped.  This
example requires a populated TreeView.  Also, the TTreeView
DragMode property must be set to dmAutomatic, and the TTreeView
OnDragOver event handler must be implemented to accept the
drop.
}
procedure TForm1.TreeView1DragDrop(Sender, Source: TObject; X, Y: Integer);
var
  AnItem: TTreeNode;
  AttachMode: TNodeAttachMode;
  HT: THitTests;
begin
  if TreeView1.Selected = nil then Exit;
  HT := TreeView1.GetHitTestInfoAt(X, Y);
  AnItem := TreeView1.GetNodeAt(X, Y);
  if (HT - [htOnItem, htOnIcon, htNowhere, htOnIndent] <> HT) then
  begin
    if (htOnItem in HT) or (htOnIcon in HT) then
      AttachMode := naAddChild
    else if htNowhere in HT then AttachMode := naAdd
    else if htOnIndent in HT then AttachMode := naInsert;
    TreeView1.Selected.MoveTo(AnItem, AttachMode);
  end;
end;

procedure TForm1.TreeView1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TTreeView;
end;
```

See Also

BeginDrag

DragCursor

EndDrag

OnDragDrop

OnDragOver

OnEndDrag

OnStartDrag

DragKind

Arranging the Order of Persistent Columns

63.1.43.4.10 TRibbonSpinButton.Enabled Property

Controls whether the control responds to mouse, keyboard, and timer events.

Pascal

```
property Enabled;
```

C++

```
__property Enabled;
```

Description

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to False. Some disabled controls appear dimmed (for example: buttons, check boxes, labels), while others (container controls) simply lose their functionality without changing their appearance. If Enabled is set to False, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to True.

Note: This property applies to all TControl descendants.

See Also

Visible

DisableControls

EnableControls

Enabling Mouse

63.1.43.4.11 TRibbonSpinButton.FocusControl Property

Pascal

```
property FocusControl: TWinControl;
```

C++

```
__property TWinControl * FocusControl;
```

Description

This is FocusControl, a member of class TRibbonSpinButton.

63.1.43.4.12 TRibbonSpinButton.ParentCtl3D Property

Determines where a component looks to determine if it should appear three dimensional.

Pascal

```
property ParentCtl3D;
```

C++

```
__property ParentCtl3D;
```

Description

ParentCtl3D is provided for backwards compatibility. It has no effect on 32-bit versions of Windows or NT 4.0 and later.

ParentCtl3D determines whether the control uses its parent's Ctl3D property.

See Also

Ctl3D

63.1.43.4.13 TRibbonSpinButton.ParentShowHint Property

Determines where a control looks to find out if its Help Hint should be shown.

Pascal

```
property ParentShowHint;
```

C++

```
__property ParentShowHint;
```

Description

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the control uses the ShowHint property value of its parent. If ParentShowHint is false, the control uses the value of its own ShowHint property.

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note: Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See Also

Hint

Parent

ParentColor

ParentFont

ShowHint

63.1.43.4.14 TRibbonSpinButton.PopupMenu Property

Identifies the pop-up menu associated with the control.

Pascal

```
property PopupMenu;
```

C++

```
__property PopupMenu;
```

Description

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the control and clicks the right mouse button. If the TPopupMenu's AutoPopup property is true, the pop-up menu appears automatically. If the menu's AutoPopup property is false, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

C++ Examples:

```
/*
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  PopupMenu1->AutoPopup = true;
  Edit1->PopupMenu = PopupMenu1;
  Edit2->PopupMenu = PopupMenu1;
}

/*
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
*/
void __fastcall TForm1::Copy1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CopyToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Cut1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CutToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Paste1Click(TObject *Sender)
```

```
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->PasteFromClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}
/*
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
    {
        switch (ClickedItem->Tag)
        {
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
            }
        }
    }
}
```

```
/*
To edit or destroy an item, grab its pointer via the Items
property.
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}

void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  PopupMenu1.AutoPopup := True;
  Edit1.PopupMenu := PopupMenu1;
  Edit2.PopupMenu := PopupMenu1;
end;

{
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
}
procedure TForm1.Copy1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
   Edit1.CopyToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CopyToClipboard
  else
    Beep;
end;
```

```
procedure TForm1.Cut1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.CutToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CutToClipboard
  else
    Beep;
end;

procedure TForm1.Paste1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.PasteFromClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.PasteFromClipboard
  else
    Beep;
end;
{
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
}

type
  TForm1 = class(TForm)
    AddButton: TButton;
    EditButton: TButton;
    DestroyButton: TButton;
    PopupMenu1: TPopupMenu;
    procedure AddButtonClick(Sender: TObject);
    procedure EditButtonClick(Sender: TObject);
    procedure DestroyButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    procedure PopupMenuItemsClick(Sender: TObject);

  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.AddButtonClick(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
    // The owner (PopupMenu1) will cleanup this menu item.
  NewItem := TMenuItem.Create(PopupMenu1); // create the new item
  index := PopupMenu1.Items.Count;
  PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
  NewItem.Caption := 'Menu Item ' + IntToStr(index);
  NewItem.Tag := index;
```

```
    NewItem.OnClick :=
      PopupMenuItemsClick; // assign it an event handler
  end;

  procedure TForm1.PopupMenuItemsClick(Sender: TObject);
  begin
    with Sender as TMenuItem do
    begin
      case Tag of
        0:  ShowMessage('first item clicked');
        1:  ShowMessage('second item clicked');
        2:  ShowMessage('third item clicked');
        3:  ShowMessage('fourth item clicked');
      end;
    end;
  end;

  {
  To edit or destroy an item, grab its pointer via the Items
  property.
  }
  procedure TForm1.EditButtonClick(Sender: TObject);
  var
    ItemToEdit: TMenuItem;
  begin
    ItemToEdit := PopupMenu.Items[1];
    ItemToEdit.Caption := 'Changed Caption';
  end;

  procedure TForm1.DestroyButtonClick(Sender: TObject);
  var
    ItemToDelete: TMenuItem;
  begin
    ItemToDelete := PopupMenu.Items[2];
    ItemToDelete.Free;
  end;

  procedure TForm1.FormCreate(Sender: TObject);
  var
    index: Integer;
    NewItem: TMenuItem;
  begin
    for index := 0 to 3 do
    begin
        // The owner (PopupMenu1) will cleanup this menu item.
      NewItem := TMenuItem.Create(PopupMenu1); // create the new item
      PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
      NewItem.Caption := 'Menu Item ' + IntToStr(index);
      NewItem.Tag := index;
      NewItem.OnClick :=
        PopupMenuItemsClick; // assign it an event handler
    end;
  end;
```

See Also

AutoPopup (see page 1437)

Popup (see page 1434)

OnContextPopup

63.1.43.4.15 TRibbonSpinButton.ShowHint Property

Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.

Pascal

```
property ShowHint;
```

C++

```
__property ShowHint;
```

Description

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either

the control's own ShowHint property must be true, or

the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

C++ Examples:

```
/*
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  Edit1->Hint = "Enter your name";
  Edit1->ShowHint = true;

  char string[10];
  char index[3];
  for(int i = 1; i <= 10; i++)
  {
    strcpy(string, "Item");
    itoa(i, index, 10);
    strcat(string, index);
    ListBox1->Items->Add(string);
  }
  ListBox1->Hint = "Select an item";
  ListBox1->ShowHint = true;
  Application->ShowHint = true;
}
/*
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
*/
```

Delphi Examples:

```
{
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
```

```
    Edit1.Hint := 'Enter your name';
    Edit1.ShowHint := True;
    with ListBox1 do
    begin
      for I := 1 to 10 do
        Items.Add('Item ' + IntToStr(I));
      Hint := 'Select an item';
      ShowHint := True;
    end;
    Application.ShowHint := True;
  end;
  {
  Note: To see an example that displays the hints of controls
  in some place other than in a Help Hint box, see the OnHint
  example.
  }
```

See Also

Hint

ParentShowHint

ShowHint

Displaying Fly-over Help

63.1.43.4.16 TRibbonSpinButton.TabOrder Property

Indicates the position of the control in its parent's tab order.

Pascal

```
property TabOrder;
```

C++

```
__property TabOrder;
```

Description

TabOrder is the order in which child windows are visited when the user presses the Tab key. The control with the TabOrder value of 0 is the control that has the focus when the form first appears.

Initially, the tab order is always the order in which the controls were added to the form. The first control added to the form has a TabOrder value of 0, the second is 1, the third is 2, and so on. Change this by changing the TabOrder property.

Each control has a unique tab-order value within its parent. If you change the TabOrder property value of one control to be the same as the value of a second control, the TabOrder value for all the other controls changes. For example, suppose a control is sixth in the tab order. If you change the control's TabOrder property value to 3 (making the control fourth in the tab order), the control that was originally fourth in the tab order now becomes fifth, and the control that was fifth becomes sixth.

Assigning TabOrder a value greater than the number of controls contained in the parent control moves the control to the end of the tab order. The control does not take on the assigned value of TabOrder, but instead is given the number that assures the control is the last in the tab order.

Note: TabOrder is meaningful only if the TabStop property is true and if the control has a parent. (The TabOrder property of a form is not used unless the form is the child of another form.) A control with a TabOrder of -1 has no parent, and therefore cannot be reached by pressing the Tab key. To remove a parented control from the Tab order, set its TabStop property to false.

See Also

FindNextControl

TabStop

63.1.43.4.17 TRibbonSpinButton.TabStop Property

Determines if the user can tab to a control.

Pascal

```
property TabStop;
```

C++

```
__property TabStop;
```

Description

Use the TabStop to allow or disallow access to the control using the Tab key.

If TabStop is true, the control is in the tab order. If TabStop is false, the control is not in the tab order and the user can't press the Tab key to move to the control.

Note: TabStop is not meaningful for a form unless the form assigns another form to be its parent.

See Also

TabOrder

63.1.43.4.18 TRibbonSpinButton.UpGlyph Property

Pascal

```
property UpGlyph: TBitmap;
```

C++

```
__property TBitmap UpGlyph;
```

Description

This is UpGlyph, a member of class TRibbonSpinButton.

63.1.43.4.19 TRibbonSpinButton.UpNumGlyphs Property

Pascal

```
property UpNumGlyphs: TNumGlyphs;
```

C++

```
__property TNumGlyphs UpNumGlyphs;
```

Description

This is UpNumGlyphs, a member of class TRibbonSpinButton.

63.1.43.4.20 TRibbonSpinButton.Visible Property

Determines whether the component appears onscreen.

Pascal

```
property Visible;
```

C++

```
__property Visible;
```

Description

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

Calling the Show method sets the control's Visible property to true. Calling the Hide method sets it to false.

C++ Examples:

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
    pPage->PageControl = PageControl1;
    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this); // The owner will clean this up.
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
```

```
      begin
        PageControl := PageControl1;
        Caption := 'TabSheet' + IntToStr(i);
        with TEdit.Create(Self) do
        begin
          Parent := PageControl1.Pages[i];
          Left := Random(PageControl1.ActivePage.ClientWidth - Width);
          Top := Random(PageControl1.ActivePage.ClientHeight - Height);
        end;
      end;
      PageControl1Change(Sender);
  end;
```

See Also

Enabled

Hide

Show

VisibleChanging

63.1.43.5 TRibbonSpinButton Events

The events of the TRibbonSpinButton class are listed here.

Published Events

	Name	Description
	OnDownClick (see page 3820)	This is OnDownClick, a member of class TRibbonSpinButton.
	OnDragDrop (see page 3820)	Occurs when the user drops an object being dragged.
	OnDragOver (see page 3823)	Occurs when the user drags an object over a control.
	OnEndDock (see page 3827)	Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.
	OnEndDrag (see page 3827)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.
	OnEnter (see page 3828)	Occurs when a control receives the input focus.
	OnExit (see page 3831)	Occurs when the input focus shifts away from one control to another.
	OnStartDock (see page 3835)	Occurs when the user begins to drag a control with a DragKind of dkDock.
	OnStartDrag (see page 3835)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.
	OnUpClick (see page 3836)	This is OnUpClick, a member of class TRibbonSpinButton.

63.1.43.5.1 TRibbonSpinButton.OnDownClick Event

Pascal

```
property OnDownClick: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnDownClick;
```

Description

This is OnDownClick, a member of class TRibbonSpinButton.

63.1.43.5.2 TRibbonSpinButton.OnDragDrop Event

Occurs when the user drops an object being dragged.

Pascal

```
property OnDragDrop;
```

C++

```
__property OnDragDrop;
```

Description

Use the OnDragDrop event handler to specify what happens when the user drops an object. The Source parameter of the OnDragDrop event is the object being dropped, and the Sender is the control the object is being dropped on. The X and Y parameters are the coordinates of the mouse positioned over the control.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}

void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
```

```
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
}
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  TForm1(Source).Left := X;
  TForm1(Source).Top := Y;
end;

procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := False;
  if (Source is TLabel) and (State = dsDragEnter) then
    (Source as TLabel).EndDrag(False);
end;
```

```
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
dropped label.
}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Not');
  ListBox1.Items.Add('In');
  ListBox1.Items.Add('Alphabetical');
  ListBox1.Items.Add('Order');
end;

// This OnDragOver event handler permits the list box to
// accept a dropped label:

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

// This OnDragDrop event handler implements the drop behavior.

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
    with Sender as TListBox do
    begin
      Font := (Source as TLabel).Font;
      Color := (Source as TLabel).Color;
      DoubleBuffered := true;
      Color := clWindow;
    end;
  end;
end;
```

See Also

BeginDrag

DragDrop

Dragging

EndDrag

OnDragOver

OnEndDrag

OnStartDrag

TDragObject

63.1.43.5.3 TRibbonSpinButton.OnDragOver Event

Occurs when the user drags an object over a control.

Pascal

```
property OnDragOver;
```

C++

```
__property OnDragOver;
```

Description

Use an OnDragOver event to signal that the control can accept a dragged object so the user can drop or dock it.

Within the OnDragOver event handler, change the Accept parameter to false to reject the dragged object. Leave Accept as true to allow the user to drop or dock the dragged object on the control.

To change the shape of the cursor, indicating that the control can accept the dragged object, change the value of the DragCursor property for the control before the OnDragOver event occurs.

The Source is the object being dragged, the Sender is the potential drop or dock site, and X and Y are screen coordinates in pixels. The State parameter specifies how the dragged object is moving over the control.

Note: Within the OnDragOver event handler, the Accept parameter defaults to true. However, if an OnDragOver event handler is not supplied, the control rejects the dragged object, as if the Accept parameter were changed to false.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}

void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
```

```
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
}
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  TForm1(Source).Left := X;
  TForm1(Source).Top := Y;
end;

procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
```

```
    State: TDragState; var Accept: Boolean);
  begin
    Accept := Source is TLabel;
  end;

  procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
    State: TDragState; var Accept: Boolean);
  begin
    Accept := False;
    if (Source is TLabel) and (State = dsDragEnter) then
      (Source as TLabel).EndDrag(False);
  end;
  {
  This code requires a list box and three labels, each with a
  different font and color.  The DragMode property for each of
  the labels is dmAutomatic.  The user can select a label and
  drag it to a list box and drop it. When the label is dropped,
  the items in the list box assume the color and font of the
  dropped label.
  }

  procedure TForm1.FormCreate(Sender: TObject);
  begin
    ListBox1.Items.Add('Not');
    ListBox1.Items.Add('In');
    ListBox1.Items.Add('Alphabetical');
    ListBox1.Items.Add('Order');
  end;

  // This OnDragOver event handler permits the list box to
  // accept a dropped label:

  procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
    State: TDragState; var Accept: Boolean);
  begin
    Accept := Source is TLabel;
  end;

  // This OnDragDrop event handler implements the drop behavior.

  procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
  begin
    if (Sender is TListBox) and (Source is TLabel) then
    begin
      with Sender as TListBox do
      begin
        Font := (Source as TLabel).Font;
        Color := (Source as TLabel).Color;
        DoubleBuffered := true;
        Color := clWindow;
      end;
    end;
  end;
```

See Also

DragCursor

OnDragDrop

OnEndDrag

OnStartDrag

TDragObject

DragKind

OnEndDock

OnDockDrop

OnStartDock

63.1.43.5.4 TRibbonSpinButton.OnEndDock Event

Occurs when the dragging of an object ends, either by docking the object or by canceling the dragging.

Pascal

```
property OnEndDock;
```

C++

```
__property OnEndDock;
```

Description

Use OnEndDock to specify actions or special processing that when a drag-and-dock operation stops.

See Also

DragKind

OnDragOver

OnDockDrop

OnStartDock

63.1.43.5.5 TRibbonSpinButton.OnEndDrag Event

Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.

Pascal

```
property OnEndDrag;
```

C++

```
__property OnEndDrag;
```

Description

Use the OnEndDrag event handler to specify any special processing that occurs when dragging stops.

C++ Examples:

```
/*
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
*/
void __fastcall TForm1::Label1EndDrag(TObject *Sender, TObject *Target, int X,
      int Y)
{
  if (Sender->ClassNameIs("TLabel"))
  {
    AnsiString S = (dynamic_cast<TLabel *>(Sender))->Name + " was dropped and ";
    if (Target)
      S = S + "accepted!";
    else
      S = S + "rejected.";
    Status->Caption = S;
```

```
  }
}
```

Delphi Examples:

```
{
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
}
procedure TForm1.Label1EndDrag(Sender, Target: TObject; X, Y: Integer);
var
  S: string;
begin
  S := (Sender as TLabel).Name + ' was dropped... and ';
  if Target <> nil then
    S := S + 'accepted!'
  else
    S := S + 'rejected!';
  Status.Caption := S;
end;
```

See Also

OnDragDrop

OnDragOver

OnStartDrag

DragKind

63.1.43.5.6 TRibbonSpinButton.OnEnter Event

Occurs when a control receives the input focus.

Pascal

```
property OnEnter;
```

C++

```
__property OnEnter;
```

Description

Use the OnEnter event handler to cause any special processing to occur when a control becomes active.

The OnEnter event does not occur when switching between forms or between another application and the application that includes the control.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnEnter event occurs for the container before the OnEnter event of the contained control.

Similarly, an OnExit event of the container occurs after the OnExit event of the control in a container when focus moves to another control outside the container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

C++ Examples:

```
/*
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
*/
TWinControl *myActiveControl;

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  FontDialog1->Options << fdApplyButton;
  FontDialog1->Execute();
}

void __fastcall TForm1::FontDialog1Apply(TObject *Sender, HWND Wnd)
{
  if (myActiveControl->ClassNameIs("TEdit"))
   (dynamic_cast<TEdit *>(myActiveControl))->Font->Assign(FontDialog1->Font);
  else if (myActiveControl->ClassNameIs("TRichEdit"))
    (dynamic_cast<TRichEdit *>(myActiveControl))->SelAttributes->Assign(FontDialog1->Font);
  else
    MessageBeep(0);
}

void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  myActiveControl = ActiveControl;
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
```

```
  applied to the active control. Clicking the button sets the
  ActiveControl to the button.  That is why we need to save
  the ActiveControl in a shared OnEnter procedure.  In the rich
  edit, only the text selected converts.
  }
  procedure TForm1.Button1Click(Sender: TObject);
  begin
     FontDialog1.Options := FontDialog1.Options + [fdApplyButton];
     FontDialog1.Execute;
  end;

  procedure TForm1.Edit1Enter(Sender: TObject);
  begin
    myActiveControl := ActiveControl;
  end;

  procedure TForm1.FontDialog1Apply(Sender: TObject; Wnd: HWND);
  begin
    if myActiveControl is TEdit then
      with myActiveControl as TEdit do
         Font.Assign(TFontDialog(Sender).Font)
    else if myActiveControl is TRichEdit then
      with myActiveControl as TRichEdit do
        SelAttributes.Assign(TFontDialog(Sender).Font)
    else
      Beep;
  end;
  {
  This example uses an edit box and a memo control on a form.
  When either the edit box or the memo is the active control,
  it is colored yellow. When the active control becomes
  inactive, the color of the control returns to the Windows
  system color for a window.  These event handlers could also
  be shared.
  }
  procedure TForm1.Edit1Enter(Sender: TObject);
  begin
    Edit1.Color := clYellow;
  end;

  procedure TForm1.Edit1Exit(Sender: TObject);
  begin
    Edit1.Color := clWindow;
  end;

  procedure TForm1.Memo1Enter(Sender: TObject);
  begin
    Memo1.Color := clYellow;
  end;

  procedure TForm1.Memo1Exit(Sender: TObject);
  begin
    Memo1.Color := clWindow;
  end;
```

See Also

OnExit

ActiveControl

OnActivate

Using a Single Navigator for Multiple Datasets

63.1.43.5.7 TRibbonSpinButton.OnExit Event

Occurs when the input focus shifts away from one control to another.

Pascal

```
property OnExit;
```

C++

```
__property OnExit;
```

Description

Use the OnExit event handler to provide special processing when the control ceases to be active.

The OnExit event does not occur when switching between forms or between another application and your application.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnExit event occurs for the control inside the container before the OnExit event of the container.

Similarly, an OnEnter event of the container occurs before the OnEnter event of the control in a container when focus moves to a control inside a container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

Note: In some control classes the ActiveControl property updates before the OnExit event occurs.

C++ Examples:

```
/*
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255 ,
as this validation test cannot be accomplished
using the edit mask property alone.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  // initialize the maskedit boxes to contain the empty string
  MaskEdit1->Text = "";
  MaskEdit2->Text = "";
  MaskEdit3->Text = "";

  // use the Courier font in the maskedit boxes
  MaskEdit1->Font->Name = "Courier";
  MaskEdit2->Font->Name = "Courier";
  MaskEdit3->Font->Name = "Courier";

  // initialize the mask of each maskedit box
  // according to IP address coding conventions
```

```
  MaskEdit1->EditMask = "!099.099.099.099;1; ";
  MaskEdit2->EditMask = "!099.099.099.099;1; ";
  MaskEdit3->EditMask = "!099.099.099.099;1; ";

  MaskEdit2->OnExit = MaskEdit2Exit;
}

void __fastcall TForm1::MaskEdit2Exit(TObject *Sender)
{
  // extract the net and host address from the IP
  String IP = MaskEdit1->Text;
  int net1 = IP.SubString(0, 3).TrimRight().ToInt();
  int net2 = IP.SubString(5, 3).TrimRight().ToInt();
  int host1 = IP.SubString(9, 3).TrimRight().ToInt();
  int host2 = IP.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (net1 < 0 || net1 > 255 ||
      net2 < 0 || net2 > 255 ||
      host1 < 0 || host1 > 255 ||
      host2 < 0 || host2 > 255)
    throw(Exception("Not a valid IP address."));

  // extract the net and host mask from the subnet mask
  String mask = MaskEdit2->Text;
  int netmask1 = mask.SubString(0, 3).TrimRight().ToInt();
  int netmask2 = mask.SubString(5, 3).TrimRight().ToInt();
  int hostmask1 = mask.SubString(9, 3).TrimRight().ToInt();
  int hostmask2 = mask.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (netmask1 < 0 || netmask1 > 255 ||
      netmask2 < 0 || netmask2 > 255 ||
      hostmask1 < 0 || hostmask1 > 255 ||
      hostmask2 < 0 || hostmask2 > 255)
    throw(Exception("Not a valid subnet mask."));

  // compute the subnet address using bit-wise AND
  int sub_net1 = net1 & netmask1;
  int sub_net2 = net2 & netmask2;
  int sub_host1 = host1 & hostmask1;
  int sub_host2 = host2 & hostmask2;

  // display the subnet address
  MaskEdit3->Text =
    String(sub_net1) + "." + String(sub_net2) + "." +
    String(sub_host1) + "." + String(sub_host2);
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255,
as this validation test cannot be accomplished
using the edit mask property alone.
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  { initialize the maskedit boxes to contain the empty string }
  MaskEdit1.Text := '';
  MaskEdit2.Text := '';
  MaskEdit3.Text := '';

  { use the Courier font in the maskedit boxes }
  MaskEdit1.Font.Name := 'Courier';
  MaskEdit2.Font.Name := 'Courier';
  MaskEdit3.Font.Name := 'Courier';

  {
  initialize the mask of each maskedit box
  according to IP address coding conventions
  }
  MaskEdit1.EditMask := '!099.099.099.099;1; ';
  MaskEdit2.EditMask := '!099.099.099.099;1; ';
  MaskEdit3.EditMask := '!099.099.099.099;1; ';

  MaskEdit2.OnExit := MaskEdit2Exit;
end;

procedure TForm1.MaskEdit2Exit(Sender: TObject);
var
  net1, net2, host1, host2,
  netmask1, netmask2, hostmask1, hostmask2,
  sub_net1, sub_net2, sub_host1, sub_host2: Integer;

  IP, mask: String;

begin
  // extract the net and host address from the IP
  IP := MaskEdit1.Text;
  net1 := StrToInt(TrimRight(Copy(IP, 0, 3)));
  net2 := StrToInt(TrimRight(Copy(IP, 5, 3)));
  host1 := StrToInt(TrimRight(Copy(IP, 9, 3)));
  host2 := StrToInt(TrimRight(Copy(IP, 13, 3)));

  // a range test that you cannot validate through edit masks
```

```
    if ((net1 < 0) Or (net1 > 255) Or
        (net2 < 0) Or (net2 > 255) Or
        (host1 < 0) Or (host1 > 255) Or
        (host2 < 0) Or (host2 > 255)) then
      raise EArgumentException.Create('Not a valid IP address.');

    // extract the net and host mask from the subnet mask
    mask := MaskEdit2.Text;
    netmask1 := StrToInt(TrimRight(Copy(mask, 0, 3)));
    netmask2 := StrToInt(TrimRight(Copy(mask, 5, 3)));
    hostmask1 := StrToInt(TrimRight(Copy(mask, 9, 3)));
    hostmask2 := StrToInt(TrimRight(Copy(mask, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((netmask1 < 0) Or (netmask1 > 255) Or
        (netmask2 < 0) Or (netmask2 > 255) Or
        (hostmask1 < 0) Or (hostmask1 > 255) Or
        (hostmask2 < 0) Or (hostmask2 > 255)) then
      raise EArgumentException.Create('Not a valid subnet mask.');

    // compute the subnet address using bit-wise AND
    sub_net1 := net1 And netmask1;
    sub_net2 := net2 And netmask2;
    sub_host1 := host1 And hostmask1;
    sub_host2 := host2 And hostmask2;

    // display the subnet address
    MaskEdit3.Text :=
      IntToStr(sub_net1) + '.' + IntToStr(sub_net2) + '.' +
      IntToStr(sub_host1) + '.' + IntToStr(sub_host2);
  end;
  {
  This example uses an edit box and a memo control on a form.
  When either the edit box or the memo is the active control,
  it is colored yellow. When the active control becomes
  inactive, the color of the control returns to the Windows
  system color for a window.  These event handlers could also
  be shared.
  }
  procedure TForm1.Edit1Enter(Sender: TObject);
  begin
    Edit1.Color := clYellow;
  end;

  procedure TForm1.Edit1Exit(Sender: TObject);
  begin
    Edit1.Color := clWindow;
  end;

  procedure TForm1.Memo1Enter(Sender: TObject);
  begin
    Memo1.Color := clYellow;
  end;

  procedure TForm1.Memo1Exit(Sender: TObject);
  begin
    Memo1.Color := clWindow;
  end;
```

See Also

OnEnter

ActiveControl

63.1.43.5.8 TRibbonSpinButton.OnStartDock Event

Occurs when the user begins to drag a control with a DragKind of dkDock.

Pascal

```
property OnStartDock;
```

C++

```
__property OnStartDock;
```

Description

Use the OnStartDock event handler to implement special processing when the user starts a drag-and-dock operation by dragging the control.

The OnStartDock event handler can create a TDragDockObjectEx object for the DragObject parameter to specify the appearance of the dragging rectangle and how the dragged control interacts with potential docking sites. If you return TDragDockObjectEx as the drag object, there is no need to call the Free method for the DragObject when dragging is over. If you use TDragDockObject, your application is responsible for freeing the drag object.

If the OnStartDock event handler sets the DragObject parameter to nil (Delphi) or NULL (C++), a TDragDockObject object is automatically created.

Note: This information applies to Win32 applications only.

See Also

OnEndDock

TDragDockObjectEx

DragKind

OnDragOver

OnDockDrop

63.1.43.5.9 TRibbonSpinButton.OnStartDrag Event

Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

Pascal

```
property OnStartDrag;
```

C++

```
__property OnStartDrag;
```

Description

Use the OnStartDrag event handler to implement special processing when the user starts to drag the control or an object it contains. OnStartDrag only occurs if DragKind is dkDrag.

Sender is the control that is about to be dragged, or that contains the object about to be dragged.

The OnStartDrag event handler can create a TDragControlObjectEx instance for the DragObject parameter to specify the drag cursor, or, optionally, a drag image list. If you create a TDragControlObjectEx instance, there is no need to call the Free method for the DragObject when dragging is over. If you create, instead, a TDragControlObject instance, your application is responsible for freeing the drag object instance.

If the OnStartDrag event handler sets the DragObject parameter to nil (Delphi) or NULL (C++), a TDragControlObject object is automatically created and dragging begins on the control itself.

Note: On some controls, such as TRichEdit, the underlying Windows control handles internal drag operations. For these controls, there are no OnStartDrag or OnEndDrag events for drag operations within the control.

See Also

OnEndDrag

TDragControlObjectEx

DragKind

OnDragOver

OnDragDrop

63.1.43.5.10 TRibbonSpinButton.OnUpClick Event

Pascal

```
property OnUpClick: TNotifyEvent;
```

C++

```
__property TNotifyEvent OnUpClick;
```

Description

This is OnUpClick, a member of class TRibbonSpinButton.

63.1.44 TRibbonSpinEdit Class

Class Hierarchy

Pascal

```
TRibbonSpinEdit = class(TCustomEdit);
```

C++

```
class TRibbonSpinEdit : public TCustomEdit;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonSpinEdit.

63.1.44.1 TRibbonSpinEdit Members

The following tables list the members exposed by TRibbonSpinEdit.

Protected Methods

	Name	Description
	CMMouseenter (see page 3839)	This is CMMouseenter, a member of class TRibbonSpinEdit.
	CMMouseleave (see page 3840)	This is CMMouseleave, a member of class TRibbonSpinEdit.
V	CreateParams (see page 3840)	Initializes a window-creation parameter record passed in the Params parameter.
V	CreateWnd (see page 3840)	Creates a Windows control corresponding to the edit control.
V	DownClick (see page 3841)	This is DownClick, a member of class TRibbonSpinEdit.
V	IsValidChar (see page 3841)	This is IsValidChar, a member of class TRibbonSpinEdit.
V	KeyDown (see page 3842)	Respond to key press events.
V	KeyPress (see page 3842)	Respond to keyboard input.
V	SetEnabled (see page 3843)	Sets the value of the Enabled property.
V	SetParent (see page 3843)	Sets the parent of the control.
V	UpClick (see page 3844)	This is UpClick, a member of class TRibbonSpinEdit.
	WMKillFocus (see page 3844)	This is WMKillFocus, a member of class TRibbonSpinEdit.

Public Methods

	Name	Description
V	Create (see page 3838)	Creates an instance of TCustomEdit.
V	Destroy (see page 3838)	Destroys an instance of TWinControl.
V	GetChildren (see page 3841)	Calls a specified method for each child of the control.

Public Properties

	Name	Description
R	Button (see page 3847)	This is Button, a member of class TRibbonSpinEdit.

Published Events

	Name	Description
	OnChange (see page 3866)	Occurs when the text for the edit control may have changed.
	OnClick (see page 3867)	Occurs when the user clicks the control.
	OnDblClick (see page 3868)	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.
	OnDragDrop (see page 3869)	Occurs when the user drops an object being dragged.
	OnDragOver (see page 3872)	Occurs when the user drags an object over a control.
	OnEndDrag (see page 3875)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.
	OnEnter (see page 3876)	Occurs when a control receives the input focus.
	OnExit (see page 3878)	Occurs when the input focus shifts away from one control to another.
	OnKeyDown (see page 3882)	Occurs when a user presses any key while the control has focus.
	OnKeyPress (see page 3886)	Occurs when a key is pressed.
	OnKeyUp (see page 3887)	Occurs when the user releases a key that has been pressed.
	OnMouseDown (see page 3889)	Occurs when the user presses a mouse button with the mouse pointer over a control.
	OnMouseMove (see page 3891)	Occurs when the user moves the mouse pointer while the mouse pointer is over a control.
	OnMouseUp (see page 3893)	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.
	OnStartDrag (see page 3896)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

Published Properties

	Name	Description
	Anchors (see page 3845)	Specifies how the control is anchored to its parent.
	AutoSelect (see page 3846)	Determines whether all the text in the edit control is automatically selected when the control gets focus.
	AutoSize (see page 3846)	Determines whether the height of the edit control automatically resizes to accommodate the text.

	Color (see page 3847)	Specifies the background color of the control.
	Constraints (see page 3848)	Specifies the size constraints for the control.
	Ctl3D (see page 3849)	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
	DragCursor (see page 3849)	Indicates the image used to represent the mouse pointer when the control is being dragged.
	DragMode (see page 3850)	Determines how the control initiates drag-and-drop or drag-and-dock operations.
	EditorEnabled (see page 3851)	This is EditorEnabled, a member of class TRibbonSpinEdit.
	Enabled (see page 3851)	Controls whether the control responds to mouse, keyboard, and timer events.
	Font (see page 3852)	Controls the attributes of text written on or in the control.
	Increment (see page 3853)	This is Increment, a member of class TRibbonSpinEdit.
	MaxLength (see page 3853)	Specifies the maximum number of characters the user can enter into the edit control.
	MaxValue (see page 3854)	This is MaxValue, a member of class TRibbonSpinEdit.
	MinValue (see page 3854)	This is MinValue, a member of class TRibbonSpinEdit.
	ParentColor (see page 3854)	Determines where a control looks for its color information.
	ParentCtl3D (see page 3855)	Determines where a component looks to determine if it should appear three dimensional.
	ParentFont (see page 3855)	Determines where a control looks for its font information.
	ParentShowHint (see page 3855)	Determines where a control looks to find out if its Help Hint should be shown.
	PopupMenu (see page 3856)	Identifies the pop-up menu associated with the control.
	ReadOnly (see page 3861)	Determines whether the user can change the text of the edit control.
	ShowHint (see page 3861)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	TabOrder (see page 3863)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 3864)	Determines if the user can tab to a control.
	Value (see page 3864)	This is Value, a member of class TRibbonSpinEdit.
	Visible (see page 3864)	Determines whether the component appears onscreen.

63.1.44.2 TRibbonSpinEdit.Create Constructor

Creates an instance of TCustomEdit.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonSpinEdit(TComponent * AOwner);
```

Description

Call Create to create and initialize an instance of TCustomEdit.

AOwner is another component, typically the form, that is responsible for freeing the edit control. It becomes the value of the Owner property.

Derived objects that override the Create method should call the inherited Create before performing any additional initialization.

See Also

Owner

63.1.44.3 TRibbonSpinEdit.Destroy Destructor

Destroys an instance of TWinControl.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRibbonSpinEdit();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TWinControl reference is not nil, and only then calls Destroy.

See Also

System::TObject::Free

Destroy

63.1.44.4 TRibbonSpinEdit Methods

The methods of the TRibbonSpinEdit class are listed here.

Protected Methods

	Name	Description
	CMMouseenter (see page 3839)	This is CMMouseenter, a member of class TRibbonSpinEdit.
	CMMouseleave (see page 3840)	This is CMMouseleave, a member of class TRibbonSpinEdit.
V	CreateParams (see page 3840)	Initializes a window-creation parameter record passed in the Params parameter.
V	CreateWnd (see page 3840)	Creates a Windows control corresponding to the edit control.
V	DownClick (see page 3841)	This is DownClick, a member of class TRibbonSpinEdit.
V	IsValidChar (see page 3841)	This is IsValidChar, a member of class TRibbonSpinEdit.
V	KeyDown (see page 3842)	Respond to key press events.
V	KeyPress (see page 3842)	Respond to keyboard input.
V	SetEnabled (see page 3843)	Sets the value of the Enabled property.
V	SetParent (see page 3843)	Sets the parent of the control.
V	UpClick (see page 3844)	This is UpClick, a member of class TRibbonSpinEdit.
	WMKillFocus (see page 3844)	This is WMKillFocus, a member of class TRibbonSpinEdit.

Public Methods

	Name	Description
V	GetChildren (see page 3841)	Calls a specified method for each child of the control.

63.1.44.4.1 TRibbonSpinEdit.CMMouseenter Method

Pascal

```
procedure CMMouseenter(var Message: TMessage); message CM_MOUSEENTER;
```

C++

```
__fastcall CMMouseenter(TMessage Message);
```

Description

This is CMMouseenter, a member of class TRibbonSpinEdit.

63.1.44.4.2 TRibbonSpinEdit.CMMouseleave Method

Pascal

```
procedure CMMouseleave(var Message: TMessage); message CM_MOUSELEAVE;
```

C++

```
__fastcall CMMouseleave(TMessage Message);
```

Description

This is CMMouseleave, a member of class TRibbonSpinEdit.

63.1.44.4.3 TRibbonSpinEdit.CreateParams Method

Initializes a window-creation parameter record passed in the Params parameter.

Pascal

```
procedure CreateParams(var Params: TCreateParams); override;
```

C++

```
virtual __fastcall CreateParams(TCreateParams Params);
```

Description

The CreateWnd method calls CreateParams to initialize the parameters it passes to CreateWindowHandle. TCustomEdit overrides the inherited method to specify a scrollable single-line edit control that reflects the current values of the BorderStyle, HideSelection, PasswordChar, and ReadOnly properties.

See Also

BorderStyle

CreateWindowHandle

CreateWnd

HideSelection

PasswordChar

ReadOnly

63.1.44.4.4 TRibbonSpinEdit.CreateWnd Method

Creates a Windows control corresponding to the edit control.

Pascal

```
procedure CreateWnd; override;
```

C++

```
virtual __fastcall CreateWnd();
```

Description

CreateWnd is called when the edit control is created, and whenever a change to the edit control requires the window to be recreated. After calling the inherited method, CreateWnd sends the newly created window messages to enforce the MaxLength and PasswordChar properties.

See Also

CreateParams

CreateWindowHandle

DestroyWnd

MaxLength

PasswordChar

CreateWnd

63.1.44.4.5 TRibbonSpinEdit.DownClick Method

Pascal

```
procedure DownClick(Sender: TObject); virtual;
```

C++

```
virtual __fastcall DownClick(TObject * Sender);
```

Description

This is DownClick, a member of class TRibbonSpinEdit.

63.1.44.4.6 TRibbonSpinEdit.GetChildren Method

Calls a specified method for each child of the control.

Pascal

```
procedure GetChildren(Proc: TGetChildProc; Root: TComponent); override;
```

C++

```
virtual __fastcall GetChildren(TGetChildProc Proc, TComponent * Root);
```

Description

GetChildren is called by the streaming system that loads and saves components. Applications seldom need to call this routine.

GetChildren executes the callback specified by the Proc parameter for every child control listed by the Controls property that lists the Root parameter as its Owner.

Override GetChildren to limit or augment which child controls are saved with the control. When overriding GetChildren, call the procedure passed as the Proc parameter for every child control that should be saved. The Root parameter indicates the component (usually a form) that owns the control.

See Also

Controls

Owner

63.1.44.4.7 TRibbonSpinEdit.IsValidChar Method

Pascal

```
function IsValidChar(Key: Char): Boolean; virtual;
```

C++

```
virtual __fastcall Boolean IsValidChar(Char Key);
```

Description

This is IsValidChar, a member of class TRibbonSpinEdit.

63.1.44.4.8 TRibbonSpinEdit.KeyDown Method

Respond to key press events.

Pascal

```
procedure KeyDown(var Key: Word; Shift: TShiftState); override;
```

C++

```
virtual __fastcall KeyDown(Word Key, TShiftState Shift);
```

Description

When a windowed control receives a key-down message (WM_KEYDOWN) from Windows, its message handler calls the DoKeyDown method. If DoKeyDown determines that the control should, in fact, process the character, it calls KeyDown, passing the key code and shift-key state in the Key and Shift parameters, respectively.

KeyDown calls any event handler attached to the OnKeyDown event. Override KeyDown to provide other responses in addition to the event-handler call.

The Key parameter is the key on the keyboard. For non-alphanumeric keys, you must use WinAPI virtual key codes to determine the key pressed. For more information, search for virtual key codes in the Win32 Developer's Reference (WIN32.HLP).

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

Either KeyDown or the OnKeyDown event handler it calls can suppress further processing of a key by setting the Key parameter to zero.

See Also

KeyUp

OnKeyDown

DoKeyDown

DoKeyPress

DoKeyUp

63.1.44.4.9 TRibbonSpinEdit.KeyPress Method

Respond to keyboard input.

Pascal

```
procedure KeyPress(var Key: Char); override;
```

C++

```
virtual __fastcall KeyPress(Char Key);
```

Description

When a windowed control receives a key-press message (WM_CHAR) from Windows, its message handler calls the DoKeyPress method. If DoKeyPress determines that the control should, in fact, process the character, it calls KeyPress, passing

the key code in the Key parameter.

KeyPress calls any event handler attached to the OnKeyPress event. Override KeyPress to provide other responses in addition to the event-handler call.

Either KeyPress or the OnKeyPress event handler it calls can suppress further processing of a character by setting the Key parameter to zero.

Note: The Key parameter is the character represented by the key that is pressed, not a Windows virtual key code.

See Also

KeyDown

KeyUp

OnKeyPress

63.1.44.4.10 TRibbonSpinEdit.SetEnabled Method

Sets the value of the Enabled property.

Pascal

```
procedure SetEnabled(Value: Boolean); override;
```

C++

```
virtual __fastcall SetEnabled(Boolean Value);
```

Description

SetEnabled is the protected write implementation of the Enabled property. After changing the internal representation of the Enabled property, SetEnabled sends a Windows message back to the control so that it can respond to the change. When TControl receives this message, it invalidates the control so that its appearance reflects the change (for example, so that it appears grayed). Some descendants respond to this message in other ways (for example, windowed controls ensure that a disabled control does not have focus).

Override GetEnabled and SetEnabled to change the implementation of the Enabled property. For example, A descendant of TControl might change the way the Enabled property interacts with an associated action. When overriding SetEnabled, be sure to call the inherited method.

See Also

GetEnabled

Enabled

Invalidate

63.1.44.4.11 TRibbonSpinEdit.SetParent Method

Sets the parent of the control.

Pascal

```
procedure SetParent(AParent: TWinControl); override;
```

C++

```
virtual __fastcall SetParent(TWinControl * AParent);
```

Description

SetParent is the protected implementation of the Parent property. Override SetParent to execute additional code when the value

of the Parent property changes.

The AParent parameter specifies the new parent of the control.

If the control already has a parent, SetParent removes the control from that parent's list of controls by calling the parent's RemoveControl method. If AParent is not nil (Delphi) or NULL (C++), SetParent calls its InsertControl method to add the control to its list of controls.

The TControl destructor calls SetParent(nil (Delphi) or NULL (C++)) to remove the control from its parent's control list before destroying the component.

See Also

Destroy

InsertControl

Parent

RemoveControl

63.1.44.4.12 TRibbonSpinEdit.UpClick Method

Pascal

```
procedure UpClick(Sender: TObject); virtual;
```

C++

```
virtual __fastcall UpClick(TObject * Sender);
```

Description

This is UpClick, a member of class TRibbonSpinEdit.

63.1.44.4.13 TRibbonSpinEdit.WMKillFocus Method

Pascal

```
procedure WMKillFocus(var Message: TWMKillFocus); message WM_KILLFOCUS;
```

C++

```
__fastcall WMKillFocus(TWMKillFocus Message);
```

Description

This is WMKillFocus, a member of class TRibbonSpinEdit.

63.1.44.5 TRibbonSpinEdit Properties

The properties of the TRibbonSpinEdit class are listed here.

Public Properties

	Name	Description
	Button (see page 3847)	This is Button, a member of class TRibbonSpinEdit.

Published Properties

	Name	Description
	Anchors (see page 3845)	Specifies how the control is anchored to its parent.

	AutoSelect (see page 3846)	Determines whether all the text in the edit control is automatically selected when the control gets focus.
	AutoSize (see page 3846)	Determines whether the height of the edit control automatically resizes to accommodate the text.
	Color (see page 3847)	Specifies the background color of the control.
	Constraints (see page 3848)	Specifies the size constraints for the control.
	Ctl3D (see page 3849)	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
	DragCursor (see page 3849)	Indicates the image used to represent the mouse pointer when the control is being dragged.
	DragMode (see page 3850)	Determines how the control initiates drag-and-drop or drag-and-dock operations.
	EditorEnabled (see page 3851)	This is EditorEnabled, a member of class TRibbonSpinEdit.
	Enabled (see page 3851)	Controls whether the control responds to mouse, keyboard, and timer events.
	Font (see page 3852)	Controls the attributes of text written on or in the control.
	Increment (see page 3853)	This is Increment, a member of class TRibbonSpinEdit.
	MaxLength (see page 3853)	Specifies the maximum number of characters the user can enter into the edit control.
	MaxValue (see page 3854)	This is MaxValue, a member of class TRibbonSpinEdit.
	MinValue (see page 3854)	This is MinValue, a member of class TRibbonSpinEdit.
	ParentColor (see page 3854)	Determines where a control looks for its color information.
	ParentCtl3D (see page 3855)	Determines where a component looks to determine if it should appear three dimensional.
	ParentFont (see page 3855)	Determines where a control looks for its font information.
	ParentShowHint (see page 3855)	Determines where a control looks to find out if its Help Hint should be shown.
	PopupMenu (see page 3856)	Identifies the pop-up menu associated with the control.
	ReadOnly (see page 3861)	Determines whether the user can change the text of the edit control.
	ShowHint (see page 3861)	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
	TabOrder (see page 3863)	Indicates the position of the control in its parent's tab order.
	TabStop (see page 3864)	Determines if the user can tab to a control.
	Value (see page 3864)	This is Value, a member of class TRibbonSpinEdit.
	Visible (see page 3864)	Determines whether the component appears onscreen.

63.1.44.5.1 TRibbonSpinEdit.Anchors Property

Specifies how the control is anchored to its parent.

Pascal

```
property Anchors;
```

C++

```
__property Anchors;
```

Description

Use Anchors to ensure that a control maintains its current position relative to an edge of its parent, even if the parent is resized. When its parent is resized, the control holds its position relative to the edges to which it is anchored.

If a control is anchored to opposite edges of its parent, the control stretches when its parent is resized. For example, if a control has its Anchors property set to [akLeft, akRight], the control stretches when the width of its parent changes.

Anchors is enforced only when the parent is resized. Thus, for example, if a control is anchored to opposite edges of a form at design time and the form is created in a maximized state, the control is not stretched because the form is not resized after the control is created.

Note: If a control should maintain contact with three edges of its parent (hugging one side of the parent and stretching the length of that side), use the Align property instead. Unlike Anchors, Align allows controls to adjust to changes in the size of other aligned sibling controls as well as changes to the parent's size.

See Also

Parent

Align

63.1.44.5.2 TRibbonSpinEdit.AutoSelect Property

Determines whether all the text in the edit control is automatically selected when the control gets focus.

Pascal

```
property AutoSelect;
```

C++

```
__property AutoSelect;
```

Description

Set AutoSelect to select all the text when the edit control gets focus. AutoSelect only applies to single-line edit controls.

Use AutoSelect when the user is more likely to replace the text in the edit control than to append to it.

See Also

SelectAll

SelLength

SelStart

SelText

SetSelTextBuf

63.1.44.5.3 TRibbonSpinEdit.AutoSize Property

Determines whether the height of the edit control automatically resizes to accommodate the text.

Pascal

```
property AutoSize;
```

C++

```
__property AutoSize;
```

Description

Use AutoSize to make the edit control adjust its size automatically so the client area accommodates the height of the text. When AutoSize is false, the edit control has a fixed height. When AutoSize is true, the size of the control is readjusted whenever a change occurs that could affect the height of the control, such as a change to the font or border style.

Note: AutoSize only adjusts the height of the edit control. Even with AutoSize set to true, text in the edit control may appear truncated if it extends beyond the width of the control.

C++ Examples:

```
/*
This example uses an edit box and a button on a form. When
the user clicks the button, the font in the edit box
enlarges, and the edit box enlarges also to accommodate the
larger font size.
```

```
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Edit1->AutoSize = true;
  Edit1->Font->Size = 20;
  Edit1->Text = "The edit box is taller now";
}
```

Delphi Examples:

```
{
{
This example uses an edit box and a button on a form. When
the user clicks the button, the font in the edit box
enlarges, and the edit box enlarges also to accommodate the
larger font size.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Edit1.AutoSize := True;
  Edit1.Font.Size := 20;
  Edit1.Text := 'The edit box is taller now';
end;
```

See Also

BorderStyle

ClientHeight

ControlStyle

Height

63.1.44.5.4 TRibbonSpinEdit.Button Property

Pascal

```
property Button: TRibbonSpinButton;
```

C++

```
__property TRibbonSpinButton Button;
```

Description

This is Button, a member of class TRibbonSpinEdit.

63.1.44.5.5 TRibbonSpinEdit.Color Property

Specifies the background color of the control.

Pascal

```
property Color;
```

C++

```
__property Color;
```

Description

Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to false.

C++ Examples:

```
/*
The following code changes the color of a shape control
using the Color dialog box. The example displays the Color
dialog box when the Button1 button is clicked, allowing the
user to select a color with the dialog box. When a color
value is selected with the dialog box, it is assigned to
the shape control.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  if (ColorDialog1->Execute())
    Shape1->Brush->Color = ColorDialog1->Color;
}
```

Delphi Examples:

```
{
The following code changes the color of a shape control
using the Color dialog box. The example displays the Color
dialog box when the Button1 button is clicked, allowing the
user to select a color with the dialog box. When a color
value is selected with the dialog box, it is assigned to
the shape control.
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  if ColorDialog1.Execute then
    Shape1.Brush.Color := ColorDialog1.Color;
end;
```

See Also

ParentColor

63.1.44.5.6 TRibbonSpinEdit.Constraints Property

Specifies the size constraints for the control.

Pascal

```
property Constraints;
```

C++

```
__property Constraints;
```

Description

Use Constraints to specify the minimum and maximum width and height of the control. When Constraints contains maximum or minimum values, the control cannot be resized to violate those constraints.

Note: Constraints apply to the height of the form. However, the height of the form depends on how large Windows makes the title bar. For example, Windows XP uses much larger title bars than other windowing systems. To work around this, note the ClientHeight when you design your form and set the constraints in the FormCreate event. In the following example, ClientHeight is represented by x: `Constraints.MinHeight := x + Height - ClientHeight`.

Warning: Do not set up constraints that conflict with the value of the Align or Anchors property. When these properties conflict, the response of the control to resize attempts is not well-defined.

See Also

AutoSize

Align

Anchors

OnConstrainedResize

OnCanResize

63.1.44.5.7 TRibbonSpinEdit.Ctl3D Property

Determines whether a control has a three-dimensional (3-D) or two-dimensional look.

Pascal

```
property Ctl3D;
```

C++

```
__property Ctl3D;
```

Description

Ctl3D is provided for backward compatibility. It is not used by 32-bit versions of Windows or NT4.0 and later, with the exception of Windows XP Home, where it is used on occasion.

On earlier platforms, Ctl3D controlled whether the control had a flat or beveled appearance.

See Also

ParentCtl3D

BevelKind

BevelInner

BevelOuter

BorderWidth

63.1.44.5.8 TRibbonSpinEdit.DragCursor Property

Indicates the image used to represent the mouse pointer when the control is being dragged.

Pascal

```
property DragCursor;
```

C++

```
__property DragCursor;
```

Description

Use the DragCursor property to change the cursor image presented when the control is being dragged.

Note: To make a custom cursor available for the DragCursor property, see the Cursor property.

See Also

Cursor

DragKind

DragMode

63.1.44.5.9 TRibbonSpinEdit.DragMode Property

Determines how the control initiates drag-and-drop or drag-and-dock operations.

Pascal

```
property DragMode;
```

C++

```
__property DragMode;
```

Description

Use DragMode to control when the user can drag the control. Disable the drag-and-drop or drag-and-dock capability at runtime by setting the DragMode property value to dmManual. Enable automatic dragging by setting DragMode to dmAutomatic.

C++ Examples:

```
/*
The following code uses GetNodeAt to add a dragged node as a
child of the node under the mouse when it is dropped.  This
example requires a populated TreeView.  Also, the TTreeView
DragMode property set to dmAutomatic, and the TTreeView
OnDragOver event handler must be implemented to accept the
drop.
*/
void __fastcall TForm1::TreeView1DragDrop(TObject *Sender, TObject *Source,
      int X, int Y)
{
  if (Source != dynamic_cast<TObject *>(TreeView1) || TreeView1->Selected == NULL)
    return;
  THitTests HT = TreeView1->GetHitTestInfoAt(X, Y);
  TNodeAttachMode AttachMode;
  TTreeNode *Item = TreeView1->GetNodeAt(X, Y);
  if (HT.Contains(htOnItem) || HT.Contains(htOnIcon))
    AttachMode = naAddChild;
  else if (HT.Contains(htNowhere))
    AttachMode = naAdd;
  else if (HT.Contains(htOnIndent))
    AttachMode = naInsert;
  else
    return;
  TreeView1->Selected->MoveTo(Item, AttachMode);
}

void __fastcall TForm1::TreeView1DragOver(TObject *Sender, TObject *Source,
      int X, int Y, TDragState State, bool &Accept)
{
  Accept = (Source == dynamic_cast<TObject *>(TreeView1));
}
```

Delphi Examples:

```
{
The following code uses GetNodeAt to add a dragged node as a
child of the node under the mouse when it is dropped.  This
example requires a populated TreeView.  Also, the TTreeView
DragMode property must be set to dmAutomatic, and the TTreeView
OnDragOver event handler must be implemented to accept the
drop.
}
procedure TForm1.TreeView1DragDrop(Sender, Source: TObject; X, Y: Integer);
var
  AnItem: TTreeNode;
  AttachMode: TNodeAttachMode;
```

```
    HT: THitTests;
  begin
    if TreeView1.Selected = nil then Exit;
    HT := TreeView1.GetHitTestInfoAt(X, Y);
    AnItem := TreeView1.GetNodeAt(X, Y);
    if (HT - [htOnItem, htOnIcon, htNowhere, htOnIndent] <> HT) then
    begin
      if (htOnItem in HT) or (htOnIcon in HT) then
        AttachMode := naAddChild
      else if htNowhere in HT then AttachMode := naAdd
      else if htOnIndent in HT then AttachMode := naInsert;
      TreeView1.Selected.MoveTo(AnItem, AttachMode);
    end;
  end;

  procedure TForm1.TreeView1DragOver(Sender, Source: TObject; X, Y: Integer;
    State: TDragState; var Accept: Boolean);
  begin
    Accept := Source is TTreeView;
  end;
```

See Also

BeginDrag

DragCursor

EndDrag

OnDragDrop

OnDragOver

OnEndDrag

OnStartDrag

DragKind

Arranging the Order of Persistent Columns

63.1.44.5.10 TRibbonSpinEdit.EditorEnabled Property

Pascal

```
property EditorEnabled: Boolean;
```

C++

```
__property Boolean EditorEnabled;
```

Description

This is EditorEnabled, a member of class TRibbonSpinEdit.

63.1.44.5.11 TRibbonSpinEdit.Enabled Property

Controls whether the control responds to mouse, keyboard, and timer events.

Pascal

```
property Enabled;
```

C++

```
__property Enabled;
```

Description

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to False. Some disabled controls appear dimmed (for example: buttons, check boxes, labels), while others (container controls) simply lose their functionality without changing their appearance. If Enabled is set to False, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to True.

Note: This property applies to all TControl descendants.

See Also

Visible

DisableControls

EnableControls

Enabling Mouse

63.1.44.5.12 TRibbonSpinEdit.Font Property

Controls the attributes of text written on or in the control.

Pascal

```
property Font;
```

C++

```
__property Font;
```

Description

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

C++ Examples:

```
/*
To use this example, place a TComboBox and TRichEdit or
TMemo on a form.  During form creation, a list of font names
is loaded into the combo box. When the combo box is clicked,
the memo or Rich Edit control font is set to the
corresponding font name in the combo box.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < Screen->Fonts->Count; i++)
   ComboBox1->Items->Add(Screen->Fonts->Strings[i]);
//  ComboBox1->Items = Screen->Fonts;  // Another way to do it.
}

void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
  RichEdit1->Font->Name = ComboBox1->Text;
}
```

Delphi Examples:

```
{
To use this example, place a TComboBox and TRichEdit or
TMemo on a form.  During form creation, a list of font names
is loaded into the combo box. When the combo box is clicked,
the memo or Rich Edit control font is set to the
```

```
  corresponding font name in the combo box.
  }
  procedure TForm1.FormCreate(Sender: TObject);
  var
    i : Integer;
  begin
    for i := 0 to Screen.Fonts.Count - 1 do
     ComboBox1.Items.Add(Screen.Fonts.Strings[i]);
  //  ComboBox1.Items := Screen.Fonts;  // Another way to do it.
    RichEdit1.Lines.Text := 'How is it to be you?';
  end;

  procedure TForm1.ComboBox1Click(Sender: TObject);
  begin
    RichEdit1.Font.Name := ComboBox1.Text;
  end;
```

See Also

DesktopFont

ParentFont

Displaying and Editing Text in a Memo Control

63.1.44.5.13 TRibbonSpinEdit.Increment Property

Pascal

```
property Increment: LongInt;
```

C++

```
__property LongInt Increment;
```

Description

This is Increment, a member of class TRibbonSpinEdit.

63.1.44.5.14 TRibbonSpinEdit.MaxLength Property

Specifies the maximum number of characters the user can enter into the edit control.

Pascal

```
property MaxLength;
```

C++

```
__property MaxLength;
```

Description

Use MaxLength to limit the number of characters that can be entered into the edit control. A value of 0 indicates that there is no application-defined limit on the length.

Use MaxLength to limit the length of the text in an edit control if that text will be copied into a fixed-length buffer.

Note: Setting MaxLength will not truncate the existing text, it merely prevents the user from adding more text after reaching the limit of MaxLength characters.

Note: Even when MaxLength is 0, there may be limitations imposed by the operating system on the number of characters that may be entered into an edit control.

Note: The unit of MaxLength is characters (not bytes) if the application is using themes.

See Also

Text

Displaying and Editing Text in a Memo Control

Displaying and Editing Text in a Rich Edit Memo Control

63.1.44.5.15 TRibbonSpinEdit.MaxValue Property

Pascal

```
property MaxValue: LongInt;
```

C++

```
__property LongInt MaxValue;
```

Description

This is MaxValue, a member of class TRibbonSpinEdit.

63.1.44.5.16 TRibbonSpinEdit.MinValue Property

Pascal

```
property MinValue: LongInt;
```

C++

```
__property LongInt MinValue;
```

Description

This is MinValue, a member of class TRibbonSpinEdit.

63.1.44.5.17 TRibbonSpinEdit.ParentColor Property

Determines where a control looks for its color information.

Pascal

```
property ParentColor;
```

C++

```
__property ParentColor;
```

Description

To have a control use the same color as its parent control, set ParentColor to true. If ParentColor is false, the control uses its own Color property.

Set ParentColor to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentColor is true for all controls in a form, changing the background color of the form to gray causes all the controls on the form to also have a gray background.

When the value of a control's Color property changes, ParentColor becomes false automatically.

See Also

Color

Parent

ParentFont

ParentShowHint

63.1.44.5.18 TRibbonSpinEdit.ParentCtl3D Property

Determines where a component looks to determine if it should appear three dimensional.

Pascal

```
property ParentCtl3D;
```

C++

```
__property ParentCtl3D;
```

Description

ParentCtl3D is provided for backwards compatibility. It has no effect on 32-bit versions of Windows or NT 4.0 and later.

ParentCtl3D determines whether the control uses its parent's Ctl3D property.

See Also

Ctl3D

63.1.44.5.19 TRibbonSpinEdit.ParentFont Property

Determines where a control looks for its font information.

Pascal

```
property ParentFont;
```

C++

```
__property ParentFont;
```

Description

To have a control use the same font as its parent control, set ParentFont to true. If ParentFont is false, the control uses its own Font property. Many controls default ParentFont to true so that all the controls in a form or other container present a uniform appearance. When the value of a control's Font property changes, ParentFont becomes false automatically.

When ParentFont is true for a form, the form uses the default font.

See Also

Font

Parent

ParentColor

ParentShowHint

63.1.44.5.20 TRibbonSpinEdit.ParentShowHint Property

Determines where a control looks to find out if its Help Hint should be shown.

Pascal

```
property ParentShowHint;
```

C++

```
__property ParentShowHint;
```

Description

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the control uses the ShowHint property value of its parent. If ParentShowHint is false, the control uses the value of its own ShowHint property.

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note: Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See Also

Hint

Parent

ParentColor

ParentFont

ShowHint

63.1.44.5.21 TRibbonSpinEdit.PopupMenu Property

Identifies the pop-up menu associated with the control.

Pascal

```
property PopupMenu;
```

C++

```
__property PopupMenu;
```

Description

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the control and clicks the right mouse button. If the TPopupMenu's AutoPopup property is true, the pop-up menu appears automatically. If the menu's AutoPopup property is false, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

C++ Examples:

```
/*
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  PopupMenu1->AutoPopup = true;
  Edit1->PopupMenu = PopupMenu1;
  Edit2->PopupMenu = PopupMenu1;
}

/*
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
```

```
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
*/
void __fastcall TForm1::Copy1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CopyToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Cut1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->CutToClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}

void __fastcall TForm1::Paste1Click(TObject *Sender)
{
  TComponent *pComponent = PopupMenu1->PopupComponent;
  if (pComponent)
  {
    if (pComponent->ClassNameIs("TEdit"))
      (dynamic_cast<TEdit *>(pComponent))->PasteFromClipboard();
    else
      MessageBeep(0);
  }
  else
    MessageBeep(0);
}
/*
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
*/
void __fastcall TForm1::AddButtonClick(TObject *Sender)
{
    // The owner (PopupMenu1) will cleanup this menu item.
    TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
    int index = PopupMenu1->Items->Count;
    PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
    NewItem->Caption = "Menu Item " + IntToStr(index);
    NewItem->Tag = index;
    NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
```

```
    TNotifyEvent();
}

void __fastcall TForm1::PopupMenuItemsClick(TObject *Sender)
{
    TMenuItem *ClickedItem = dynamic_cast<TMenuItem *>(Sender);
    if (ClickedItem)
    {
        switch (ClickedItem->Tag)
        {
            case 0:
            {
                ShowMessage("first item clicked");
                break;
            }
            case 1:
            {
                ShowMessage("second item clicked");
                break;
            }
            case 2:
            {
                ShowMessage("third item clicked");
                break;
            }
            case 3:
            {
                ShowMessage("fourth item clicked");
                break;
            }
        }
    }
}

/*
To edit or destroy an item, grab its pointer via the Items
property.
*/
void __fastcall TForm1::EditButtonClick(TObject *Sender)
{
    const int index = 1;
    TMenuItem *ItemToEdit = PopupMenu->Items->Items[index];
    ItemToEdit->Caption = "Changed Caption";
}

void __fastcall TForm1::DestroyButtonClick(TObject *Sender)
{
    const int index = 2;
    TMenuItem *ItemToDelete = PopupMenu->Items->Items[index];
    delete ItemToDelete;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    const int num_items = 4;
    for (int index = 0; index < num_items; ++index)
    {
        // The owner (PopupMenu1) will cleanup this menu item.
        TMenuItem *NewItem = new TMenuItem(PopupMenu1); // create the new item
        PopupMenu1->Items->Add(NewItem);// add it to the Popupmenu
        NewItem->Caption = "Menu Item " + IntToStr(index);
        NewItem->Tag = index;
        NewItem->OnClick = PopupMenuItemsClick;// assign it an event handler
        TNotifyEvent();
    }
}
```

Delphi Examples:

```
{
This example uses two edit boxes and one pop-up menu on a form.
The pop-up menu contains Cut, Copy, and Paste commands. This code
makes the pop-up menu available to both edit boxes:
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  PopupMenu1.AutoPopup := True;
  Edit1.PopupMenu := PopupMenu1;
  Edit2.PopupMenu := PopupMenu1;
end;

{
These are the cut, copy, and paste OnClick events for the commands
on the pop-up menu. Double click on the pop-up menu and add three
TMenuItems.  Then set the captions to "Cut", "Copy" and "Paste".
The names will be automatically set to "Cut1", "Copy1", and
"Paste1".  Then double click on the OnClick events and they will
be named "Cut1Click", "Copy1Click" and "Paste1Click" respectively.
Note: only the selected portion of the string in the edit box will
be copied, cut or replaced.
}
procedure TForm1.Copy1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
   Edit1.CopyToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CopyToClipboard
  else
    Beep;
end;

procedure TForm1.Cut1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.CutToClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.CutToClipboard
  else
    Beep;
end;

procedure TForm1.Paste1Click(Sender: TObject);
begin
  if PopupMenu1.PopupComponent = Edit1 then
    Edit1.PasteFromClipboard
  else if PopupMenu1.PopupComponent = Edit2 then
    Edit2.PasteFromClipboard
  else
    Beep;
end;
{
This example demonstrates how to add and delete menu items
to a popup menu at runtime and assign an event handler to
the OnClick event.  Place a TPopupMenu and three buttons on
the form named "AddButton", "EditButton",  and
"DestroyButton" and add OnClick events to all three buttons.
Put the TPopupMenu in the PopupMenu property of the form.
Place the PopupMenuItemsClick procedure in the TForm1 type
declaration so that it can be used as the method call for
the menu item OnClick event.
}

type
```

```
  TForm1 = class(TForm)
    AddButton: TButton;
    EditButton: TButton;
    DestroyButton: TButton;
    PopupMenu1: TPopupMenu;
    procedure AddButtonClick(Sender: TObject);
    procedure EditButtonClick(Sender: TObject);
    procedure DestroyButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    procedure PopupMenuItemsClick(Sender: TObject);

  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.AddButtonClick(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
    // The owner (PopupMenu1) will cleanup this menu item.
  NewItem := TMenuItem.Create(PopupMenu1); // create the new item
  index := PopupMenu1.Items.Count;
  PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
  NewItem.Caption := 'Menu Item ' + IntToStr(index);
  NewItem.Tag := index;
  NewItem.OnClick :=
    PopupMenuItemsClick; // assign it an event handler
end;

procedure TForm1.PopupMenuItemsClick(Sender: TObject);
begin
  with Sender as TMenuItem do
  begin
    case Tag of
      0:  ShowMessage('first item clicked');
      1:  ShowMessage('second item clicked');
      2:  ShowMessage('third item clicked');
      3:  ShowMessage('fourth item clicked');
    end;
  end;
end;

{
To edit or destroy an item, grab its pointer via the Items
property.
}
procedure TForm1.EditButtonClick(Sender: TObject);
var
  ItemToEdit: TMenuItem;
begin
  ItemToEdit := PopupMenu.Items[1];
  ItemToEdit.Caption := 'Changed Caption';
end;

procedure TForm1.DestroyButtonClick(Sender: TObject);
var
  ItemToDelete: TMenuItem;
```

```
begin
  ItemToDelete := PopupMenu.Items[2];
  ItemToDelete.Free;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  index: Integer;
  NewItem: TMenuItem;
begin
  for index := 0 to 3 do
  begin
      // The owner (PopupMenu1) will cleanup this menu item.
    NewItem := TMenuItem.Create(PopupMenu1); // create the new item
    PopupMenu1.Items.Add(NewItem);// add it to the Popupmenu
    NewItem.Caption := 'Menu Item ' + IntToStr(index);
    NewItem.Tag := index;
    NewItem.OnClick :=
      PopupMenuItemsClick; // assign it an event handler
  end;
end;
```

See Also

AutoPopup (see page 1437)

Popup (see page 1434)

OnContextPopup

63.1.44.5.22 TRibbonSpinEdit.ReadOnly Property

Determines whether the user can change the text of the edit control.

Pascal

```
property ReadOnly;
```

C++

```
__property ReadOnly;
```

Description

To restrict the edit control to display only, set the ReadOnly property to true. Set ReadOnly to false to allow the contents of the edit control to be edited.

Setting ReadOnly to true ensures that the text is not altered, while still allowing the user to select text. The selected text can then be manipulated by the application, or copied to the Clipboard.

See Also

CopyToClipboard

GetSelTextBuf

63.1.44.5.23 TRibbonSpinEdit.ShowHint Property

Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.

Pascal

```
property ShowHint;
```

C++

```
__property ShowHint;
```

Description

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either

the control's own ShowHint property must be true, or

the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

C++ Examples:

```
/*
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  Edit1->Hint = "Enter your name";
  Edit1->ShowHint = true;

  char string[10];
  char index[3];
  for(int i = 1; i <= 10; i++)
  {
    strcpy(string, "Item");
    itoa(i, index, 10);
    strcat(string, index);
    ListBox1->Items->Add(string);
  }
  ListBox1->Hint = "Select an item";
  ListBox1->ShowHint = true;
  Application->ShowHint = true;
}
/*
Note: To see an example that displays the hints of controls
in some place other than in a Help Hint box, see the OnHint
example.
*/
```

Delphi Examples:

```
{
This example uses an edit box and a list box on a form.
Items are added to the list box and a Help Hint is assigned
to both controls. The last statement enables the Help Hints
for the entire application.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
  Edit1.Hint := 'Enter your name';
  Edit1.ShowHint := True;
  with ListBox1 do
```

```
    begin
      for I := 1 to 10 do
        Items.Add('Item ' + IntToStr(I));
      Hint := 'Select an item';
      ShowHint := True;
    end;
    Application.ShowHint := True;
  end;
  {
  Note: To see an example that displays the hints of controls
  in some place other than in a Help Hint box, see the OnHint
  example.
  }
```

See Also

Hint

ParentShowHint

ShowHint

Displaying Fly-over Help

63.1.44.5.24 TRibbonSpinEdit.TabOrder Property

Indicates the position of the control in its parent's tab order.

Pascal

```
property TabOrder;
```

C++

```
__property TabOrder;
```

Description

TabOrder is the order in which child windows are visited when the user presses the Tab key. The control with the TabOrder value of 0 is the control that has the focus when the form first appears.

Initially, the tab order is always the order in which the controls were added to the form. The first control added to the form has a TabOrder value of 0, the second is 1, the third is 2, and so on. Change this by changing the TabOrder property.

Each control has a unique tab-order value within its parent. If you change the TabOrder property value of one control to be the same as the value of a second control, the TabOrder value for all the other controls changes. For example, suppose a control is sixth in the tab order. If you change the control's TabOrder property value to 3 (making the control fourth in the tab order), the control that was originally fourth in the tab order now becomes fifth, and the control that was fifth becomes sixth.

Assigning TabOrder a value greater than the number of controls contained in the parent control moves the control to the end of the tab order. The control does not take on the assigned value of TabOrder, but instead is given the number that assures the control is the last in the tab order.

Note: TabOrder is meaningful only if the TabStop property is true and if the control has a parent. (The TabOrder property of a form is not used unless the form is the child of another form.) A control with a TabOrder of -1 has no parent, and therefore cannot be reached by pressing the Tab key. To remove a parented control from the Tab order, set its TabStop property to false.

See Also

FindNextControl

TabStop

63.1.44.5.25 TRibbonSpinEdit.TabStop Property

Determines if the user can tab to a control.

Pascal

```
property TabStop;
```

C++

```
__property TabStop;
```

Description

Use the TabStop to allow or disallow access to the control using the Tab key.

If TabStop is true, the control is in the tab order. If TabStop is false, the control is not in the tab order and the user can't press the Tab key to move to the control.

Note: TabStop is not meaningful for a form unless the form assigns another form to be its parent.

See Also

TabOrder

63.1.44.5.26 TRibbonSpinEdit.Value Property

Pascal

```
property Value: LongInt;
```

C++

```
__property LongInt Value;
```

Description

This is Value, a member of class TRibbonSpinEdit.

63.1.44.5.27 TRibbonSpinEdit.Visible Property

Determines whether the component appears onscreen.

Pascal

```
property Visible;
```

C++

```
__property Visible;
```

Description

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

Calling the Show method sets the control's Visible property to true. Calling the Hide method sets it to false.

C++ Examples:

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
```

```
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
    pPage->PageControl = PageControl1;
    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this); // The owner will clean this up.
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
    begin
      PageControl := PageControl1;
      Caption := 'TabSheet' + IntToStr(i);
      with TEdit.Create(Self) do
      begin
        Parent := PageControl1.Pages[i];
        Left := Random(PageControl1.ActivePage.ClientWidth - Width);
        Top := Random(PageControl1.ActivePage.ClientHeight - Height);
      end;
    end;
    PageControl1Change(Sender);
end;
```

See Also

Enabled

Hide

Show

VisibleChanging

63.1.44.6 TRibbonSpinEdit Events

The events of the TRibbonSpinEdit class are listed here.

Published Events

	Name	Description
	OnChange (see page 3866)	Occurs when the text for the edit control may have changed.
	OnClick (see page 3867)	Occurs when the user clicks the control.
	OnDblClick (see page 3868)	Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.
	OnDragDrop (see page 3869)	Occurs when the user drops an object being dragged.
	OnDragOver (see page 3872)	Occurs when the user drags an object over a control.
	OnEndDrag (see page 3875)	Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.
	OnEnter (see page 3876)	Occurs when a control receives the input focus.
	OnExit (see page 3878)	Occurs when the input focus shifts away from one control to another.
	OnKeyDown (see page 3882)	Occurs when a user presses any key while the control has focus.
	OnKeyPress (see page 3886)	Occurs when a key is pressed.
	OnKeyUp (see page 3887)	Occurs when the user releases a key that has been pressed.
	OnMouseDown (see page 3889)	Occurs when the user presses a mouse button with the mouse pointer over a control.
	OnMouseMove (see page 3891)	Occurs when the user moves the mouse pointer while the mouse pointer is over a control.
	OnMouseUp (see page 3893)	Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.
	OnStartDrag (see page 3896)	Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

63.1.44.6.1 TRibbonSpinEdit.OnChange Event

Occurs when the text for the edit control may have changed.

Pascal

```
property OnChange;
```

C++

```
__property OnChange;
```

Description

Write an OnChange event handler to take specific action whenever the text for the edit control may have changed. Use the Modified property to see if a change actually occurred. The Text property of the edit control will already be updated to reflect any changes. This event provides the first opportunity to respond to modifications that the user types into the edit control.

See Also

Change

Text

63.1.44.6.2 TRibbonSpinEdit.OnClick Event

Occurs when the user clicks the control.

Pascal

```
property OnClick;
```

C++

```
__property OnClick;
```

Description

Use the OnClick event handler to respond when the user clicks the control. If the control has an associated action, and that action has an OnExecute method, the action's OnExecute method responds to click events unless it is superseded by an OnClick event handler.

Usually OnClick occurs when the user presses and releases the left mouse button with the mouse pointer over the control. This event can also occur when:

- The user selects an item in a grid, outline, list, or combo box by pressing an arrow key.
- The user presses Spacebar while a button or check box has focus.
- The user presses Enter when the active form has a default button (specified by the Default property).
- The user presses Esc when the active form has a cancel button (specified by the Cancel property).
- The user presses the accelerator key for a button or check box. For example, if the value of the Caption property of a check box is '&Bold', the B is underlined at runtime and the OnClick event of the check box is triggered when the user presses Alt+B. However, focus does not move to the control in these instances.
- The Checked property of a radio button is set to true.
- The value of the Checked property of a check box is changed.
- The Click method of a menu item is called.

For a form, an OnClick event occurs when the user clicks a blank area of the form or on a disabled component.

C++ Examples:

```
/*
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
*/
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this); // The owner will clean this up.
    pPage->PageControl = PageControl1;
    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this); // The owner will clean this up.
```

```
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Delphi Examples:

```
{
This examples requires a TPageControl and a TCheckBox.  The
form OnCreate event handler populates each sheet with an
edit control placed somewhere on the client area of the
sheet.  Unchecking the checkbox will cause the client area
(not the tab area) of the Tab Control to become invisible.
}
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  PageControl1.ActivePage.Visible := CheckBox1.Checked;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
  CheckBox1.Checked := PageControl1.ActivePage.Visible;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  for i := 0 to 9 do
    with TTabSheet.Create(Self) do
    begin
      PageControl := PageControl1;
      Caption := 'TabSheet' + IntToStr(i);
      with TEdit.Create(Self) do
      begin
        Parent := PageControl1.Pages[i];
        Left := Random(PageControl1.ActivePage.ClientWidth - Width);
        Top := Random(PageControl1.ActivePage.ClientHeight - Height);
      end;
    end;
    PageControl1Change(Sender);
end;
```

See Also

OnDblClick

Cancel

Default

Click (see page 1375)

OnExecute

Action

Adding Silent Video Clips to an Application

63.1.44.6.3 TRibbonSpinEdit.OnDblClick Event

Occurs when the user double-clicks the left mouse button when the mouse pointer is over the control.

Pascal

```
property OnDblClick;
```

C++

```
__property OnDblClick;
```

Description

Use the OnDblClick event to respond to mouse double-clicks.

See Also

OnClick

Responding to User Actions at Runtime

63.1.44.6.4 TRibbonSpinEdit.OnDragDrop Event

Occurs when the user drops an object being dragged.

Pascal

```
property OnDragDrop;
```

C++

```
__property OnDragDrop;
```

Description

Use the OnDragDrop event handler to specify what happens when the user drops an object. The Source parameter of the OnDragDrop event is the object being dropped, and the Sender is the control the object is being dropped on. The X and Y parameters are the coordinates of the mouse positioned over the control.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}

void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
```

```
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
}
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
```

```
{$R *.dfm}

procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  TForm1(Source).Left := X;
  TForm1(Source).Top := Y;
end;

procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := False;
  if (Source is TLabel) and (State = dsDragEnter) then
    (Source as TLabel).EndDrag(False);
end;
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
dropped label.
}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Not');
  ListBox1.Items.Add('In');
  ListBox1.Items.Add('Alphabetical');
  ListBox1.Items.Add('Order');
end;

// This OnDragOver event handler permits the list box to
// accept a dropped label:

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

// This OnDragDrop event handler implements the drop behavior.

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
    with Sender as TListBox do
    begin
      Font := (Source as TLabel).Font;
      Color := (Source as TLabel).Color;
      DoubleBuffered := true;
      Color := clWindow;
    end;
  end;
end;
```

See Also

BeginDrag

DragDrop

Dragging

EndDrag

OnDragOver

OnEndDrag

OnStartDrag

TDragObject

63.1.44.6.5 TRibbonSpinEdit.OnDragOver Event

Occurs when the user drags an object over a control.

Pascal

```
property OnDragOver;
```

C++

```
__property OnDragOver;
```

Description

Use an OnDragOver event to signal that the control can accept a dragged object so the user can drop or dock it.

Within the OnDragOver event handler, change the Accept parameter to false to reject the dragged object. Leave Accept as true to allow the user to drop or dock the dragged object on the control.

To change the shape of the cursor, indicating that the control can accept the dragged object, change the value of the DragCursor property for the control before the OnDragOver event occurs.

The Source is the object being dragged, the Sender is the potential drop or dock site, and X and Y are screen coordinates in pixels. The State parameter specifies how the dragged object is moving over the control.

Note: Within the OnDragOver event handler, the Accept parameter defaults to true. However, if an OnDragOver event handler is not supplied, the control rejects the dragged object, as if the Accept parameter were changed to false.

C++ Examples:

```
/*
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
*/
void __fastcall TForm1::FormDragDrop(TObject *Sender, TObject *Source, int X, int Y)

{
  reinterpret_cast<TForm1 *>(Source)->Left = X;
  reinterpret_cast<TForm1 *>(Source)->Top = Y;
}

void __fastcall TForm1::FormDragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = Source->ClassNameIs("TLabel");
}
```

```
void __fastcall TForm1::Panel1DragOver(TObject *Sender, TObject *Source, int X, int Y,
          TDragState State, bool &Accept)
{
  Accept = false;
  if ((Source->ClassNameIs("TLabel")) && (State == dsDragEnter))
    dynamic_cast<TLabel *>(Source)->EndDrag(false);
}
/*
This code comes from an application that contains a list box
and three labels, each with a different font and color.  The
DragMode property for each of the labels is dmAutomatic.
The user can select a label and drag it to a list box and
drop it. When the label is dropped, the items in the list
box assume the color and font of the dropped label.
*/

// This OnDragOver event handler permits the list box to
// accept a dropped label:
void __fastcall TForm1::ListBox1DragOver(TObject *Sender, TObject *Source, int X, int Y,
TDragState State, bool &Accept)
{
   Accept = Source->ClassNameIs("TLabel");
}

/*
This OnDragDrop event handler implements the drop behavior.
*/
void __fastcall TForm1::ListBox1DragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
  if (Sender->ClassNameIs("TListBox") && Source->ClassNameIs("TLabel"))
  {
    TListBox *DestList = dynamic_cast<TListBox *>(Sender);
    DestList->Font = (dynamic_cast<TLabel *>(Source))->Font;
    DestList->Color = (dynamic_cast<TLabel *>(Source))->Color;
    DestList->DoubleBuffered = true;
    DestList->Color = clWindow;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  ListBox1->Items->Add("Not");
  ListBox1->Items->Add("In");
  ListBox1->Items->Add("Alphabetical");
  ListBox1->Items->Add("Order");

}
```

Delphi Examples:

```
{
This code is an OnDragOver event handler that won't allow a
label control to be dropped on a panel control and stops the
dragging of the label as soon as the user drags the label
onto the panel:
Note: Set the label's DragMode property to dmAutomatic.
}
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    procedure Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormDragDrop(Sender, Source: TObject; X, Y: Integer);
  private
```

```
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  TForm1(Source).Left := X;
  TForm1(Source).Top := Y;
end;

procedure TForm1.FormDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

procedure TForm1.Panel1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := False;
  if (Source is TLabel) and (State = dsDragEnter) then
    (Source as TLabel).EndDrag(False);
end;
{
This code requires a list box and three labels, each with a
different font and color.  The DragMode property for each of
the labels is dmAutomatic.  The user can select a label and
drag it to a list box and drop it. When the label is dropped,
the items in the list box assume the color and font of the
dropped label.
}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Not');
  ListBox1.Items.Add('In');
  ListBox1.Items.Add('Alphabetical');
  ListBox1.Items.Add('Order');
end;

// This OnDragOver event handler permits the list box to
// accept a dropped label:

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;

// This OnDragDrop event handler implements the drop behavior.

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
    with Sender as TListBox do
    begin
      Font := (Source as TLabel).Font;
```

```
        Color := (Source as TLabel).Color;
        DoubleBuffered := true;
        Color := clWindow;
      end;
    end;
  end;
```

See Also

DragCursor

OnDragDrop

OnEndDrag

OnStartDrag

TDragObject

DragKind

OnEndDock

OnDockDrop

OnStartDock

63.1.44.6.6 TRibbonSpinEdit.OnEndDrag Event

Occurs when the dragging of an object ends, either by dropping the object or by canceling the dragging.

Pascal

```
property OnEndDrag;
```

C++

```
__property OnEndDrag;
```

Description

Use the OnEndDrag event handler to specify any special processing that occurs when dragging stops.

C++ Examples:

```
/*
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
*/
void __fastcall TForm1::Label1EndDrag(TObject *Sender, TObject *Target, int X,
      int Y)
{
  if (Sender->ClassNameIs("TLabel"))
  {
    AnsiString S = (dynamic_cast<TLabel *>(Sender))->Name + " was dropped and ";
    if (Target)
      S = S + "accepted!";
    else
      S = S + "rejected.";
    Status->Caption = S;
  }
}
```

Delphi Examples:

```
{
```

```
This code displays a message in a label named Status. The
message displayed depends on whether the dragged label
control was successfully dropped into and accepted by a
another control.
}
procedure TForm1.Label1EndDrag(Sender, Target: TObject; X, Y: Integer);
var
  S: string;
begin
  S := (Sender as TLabel).Name + ' was dropped... and ';
  if Target <> nil then
    S := S + 'accepted!'
  else
    S := S + 'rejected!';
  Status.Caption := S;
end;
```

See Also

OnDragDrop

OnDragOver

OnStartDrag

DragKind

63.1.44.6.7 TRibbonSpinEdit.OnEnter Event

Occurs when a control receives the input focus.

Pascal

```
property OnEnter;
```

C++

```
__property OnEnter;
```

Description

Use the OnEnter event handler to cause any special processing to occur when a control becomes active.

The OnEnter event does not occur when switching between forms or between another application and the application that includes the control.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnEnter event occurs for the container before the OnEnter event of the contained control.

Similarly, an OnExit event of the container occurs after the OnExit event of the control in a container when focus moves to another control outside the container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

C++ Examples:

```
/*
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
```

```
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
*/
TWinControl *myActiveControl;

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  FontDialog1->Options << fdApplyButton;
  FontDialog1->Execute();
}

void __fastcall TForm1::FontDialog1Apply(TObject *Sender, HWND Wnd)
{
  if (myActiveControl->ClassNameIs("TEdit"))
   (dynamic_cast<TEdit *>(myActiveControl))->Font->Assign(FontDialog1->Font);
  else if (myActiveControl->ClassNameIs("TRichEdit"))
    (dynamic_cast<TRichEdit *>(myActiveControl))->SelAttributes->Assign(FontDialog1->Font);
  else
    MessageBeep(0);
}

void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  myActiveControl = ActiveControl;
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example takes an edit control, a button, and a rich
edit control on a form.  When the user presses the button, a
Font dialog appears.  When the user clicks the Apply (not OK)
button in the Font dialog, the currently selected font is
applied to the active control. Clicking the button sets the
ActiveControl to the button.  That is why we need to save
the ActiveControl in a shared OnEnter procedure.  In the rich
edit, only the text selected converts.
}
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
   FontDialog1.Options := FontDialog1.Options + [fdApplyButton];
   FontDialog1.Execute;
end;

procedure TForm1.Edit1Enter(Sender: TObject);
begin
  myActiveControl := ActiveControl;
end;

procedure TForm1.FontDialog1Apply(Sender: TObject; Wnd: HWND);
begin
  if myActiveControl is TEdit then
    with myActiveControl as TEdit do
       Font.Assign(TFontDialog(Sender).Font)
  else if myActiveControl is TRichEdit then
    with myActiveControl as TRichEdit do
      SelAttributes.Assign(TFontDialog(Sender).Font)
  else
    Beep;
end;
{
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
}
procedure TForm1.Edit1Enter(Sender: TObject);
begin
  Edit1.Color := clYellow;
end;

procedure TForm1.Edit1Exit(Sender: TObject);
begin
  Edit1.Color := clWindow;
end;

procedure TForm1.Memo1Enter(Sender: TObject);
begin
  Memo1.Color := clYellow;
end;

procedure TForm1.Memo1Exit(Sender: TObject);
begin
  Memo1.Color := clWindow;
end;
```

See Also

OnExit

ActiveControl

OnActivate

Using a Single Navigator for Multiple Datasets

63.1.44.6.8 TRibbonSpinEdit.OnExit Event

Occurs when the input focus shifts away from one control to another.

Pascal

```
property OnExit;
```

C++

```
__property OnExit;
```

Description

Use the OnExit event handler to provide special processing when the control ceases to be active.

The OnExit event does not occur when switching between forms or between another application and your application.

When switching between controls in separate container controls such as the TPanel and the TGroupBox controls, an OnExit event occurs for the control inside the container before the OnExit event of the container.

Similarly, an OnEnter event of the container occurs before the OnEnter event of the control in a container when focus moves to a control inside a container.

For example, consider a form with an OK button and a group box that contains three radio buttons, where focus is currently on the OK button. When the user clicks one of the radio buttons, an OnExit event of the button occurs, followed by an OnEnter event on the group box, and finally an OnEnter event on the radio button that was clicked. If the user then clicks on the OK button, an OnExit event for the radio button occurs followed by an OnExit event for the group box, and then the button's OnEnter event occurs.

Note: In some control classes the ActiveControl property updates before the OnExit event occurs.

C++ Examples:

```
/*
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255 ,
as this validation test cannot be accomplished
using the edit mask property alone.
*/
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  // initialize the maskedit boxes to contain the empty string
  MaskEdit1->Text = "";
  MaskEdit2->Text = "";
  MaskEdit3->Text = "";

  // use the Courier font in the maskedit boxes
  MaskEdit1->Font->Name = "Courier";
  MaskEdit2->Font->Name = "Courier";
  MaskEdit3->Font->Name = "Courier";

  // initialize the mask of each maskedit box
  // according to IP address coding conventions
  MaskEdit1->EditMask = "!099.099.099.099;1; ";
  MaskEdit2->EditMask = "!099.099.099.099;1; ";
  MaskEdit3->EditMask = "!099.099.099.099;1; ";

  MaskEdit2->OnExit = MaskEdit2Exit;
```

```
}

void __fastcall TForm1::MaskEdit2Exit(TObject *Sender)
{
  // extract the net and host address from the IP
  String IP = MaskEdit1->Text;
  int net1 = IP.SubString(0, 3).TrimRight().ToInt();
  int net2 = IP.SubString(5, 3).TrimRight().ToInt();
  int host1 = IP.SubString(9, 3).TrimRight().ToInt();
  int host2 = IP.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (net1 < 0 || net1 > 255 ||
      net2 < 0 || net2 > 255 ||
      host1 < 0 || host1 > 255 ||
      host2 < 0 || host2 > 255)
    throw(Exception("Not a valid IP address."));

  // extract the net and host mask from the subnet mask
  String mask = MaskEdit2->Text;
  int netmask1 = mask.SubString(0, 3).TrimRight().ToInt();
  int netmask2 = mask.SubString(5, 3).TrimRight().ToInt();
  int hostmask1 = mask.SubString(9, 3).TrimRight().ToInt();
  int hostmask2 = mask.SubString(13, 3).TrimRight().ToInt();

  // a range test that you cannot validate through edit masks
  if (netmask1 < 0 || netmask1 > 255 ||
      netmask2 < 0 || netmask2 > 255 ||
      hostmask1 < 0 || hostmask1 > 255 ||
      hostmask2 < 0 || hostmask2 > 255)
    throw(Exception("Not a valid subnet mask."));

  // compute the subnet address using bit-wise AND
  int sub_net1 = net1 & netmask1;
  int sub_net2 = net2 & netmask2;
  int sub_host1 = host1 & hostmask1;
  int sub_host2 = host2 & hostmask2;

  // display the subnet address
  MaskEdit3->Text =
    String(sub_net1) + "." + String(sub_net2) + "." +
    String(sub_host1) + "." + String(sub_host2);
}
/*
This example uses an edit box and a memo control on a form.
When either the edit box or the memo is the active control,
it is colored yellow. When the active control becomes
inactive, the color of the control returns to the Windows
system color for a window.  These event handlers could also
be shared.
*/
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
  Edit1->Color = clYellow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  Edit1->Color = clWindow;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
  Memo1->Color = clYellow;
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
  Memo1->Color = clWindow;
}
```

Delphi Examples:

```
{
This example uses three mask edit components on a form,
where the third mask edit box is set to be read only.
When the application runs, it initializes the mask edit boxes
to contain the empty string, to use the Courier font and to
use a particular mask, corresponding to IP address coding convetions.
The mask edit boxes display the IP address, the subnet mask
and the subnet address correspondingly. When the user exits the
subnet mask edit box, the subnet address is automatically calculated
using bit-wise AND operations and displayed in the third mask edit box.
An additional test is made to assure that the values in the IP address
and in the subnet mask are in the interval from 0 to 255,
as this validation test cannot be accomplished
using the edit mask property alone.
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  { initialize the maskedit boxes to contain the empty string }
  MaskEdit1.Text := '';
  MaskEdit2.Text := '';
  MaskEdit3.Text := '';

  { use the Courier font in the maskedit boxes }
  MaskEdit1.Font.Name := 'Courier';
  MaskEdit2.Font.Name := 'Courier';
  MaskEdit3.Font.Name := 'Courier';

  {
  initialize the mask of each maskedit box
  according to IP address coding conventions
  }
  MaskEdit1.EditMask := '!099.099.099.099;1; ';
  MaskEdit2.EditMask := '!099.099.099.099;1; ';
  MaskEdit3.EditMask := '!099.099.099.099;1; ';

  MaskEdit2.OnExit := MaskEdit2Exit;
end;

procedure TForm1.MaskEdit2Exit(Sender: TObject);
var
  net1, net2, host1, host2,
  netmask1, netmask2, hostmask1, hostmask2,
  sub_net1, sub_net2, sub_host1, sub_host2: Integer;

  IP, mask: String;

begin
  // extract the net and host address from the IP
  IP := MaskEdit1.Text;
  net1 := StrToInt(TrimRight(Copy(IP, 0, 3)));
  net2 := StrToInt(TrimRight(Copy(IP, 5, 3)));
  host1 := StrToInt(TrimRight(Copy(IP, 9, 3)));
  host2 := StrToInt(TrimRight(Copy(IP, 13, 3)));

  // a range test that you cannot validate through edit masks
  if ((net1 < 0) Or (net1 > 255) Or
      (net2 < 0) Or (net2 > 255) Or
      (host1 < 0) Or (host1 > 255) Or
      (host2 < 0) Or (host2 > 255)) then
    raise EArgumentException.Create('Not a valid IP address.');
```

```
    // extract the net and host mask from the subnet mask
    mask := MaskEdit2.Text;
    netmask1 := StrToInt(TrimRight(Copy(mask, 0, 3)));
    netmask2 := StrToInt(TrimRight(Copy(mask, 5, 3)));
    hostmask1 := StrToInt(TrimRight(Copy(mask, 9, 3)));
    hostmask2 := StrToInt(TrimRight(Copy(mask, 13, 3)));

    // a range test that you cannot validate through edit masks
    if ((netmask1 < 0) Or (netmask1 > 255) Or
        (netmask2 < 0) Or (netmask2 > 255) Or
        (hostmask1 < 0) Or (hostmask1 > 255) Or
        (hostmask2 < 0) Or (hostmask2 > 255)) then
      raise EArgumentException.Create('Not a valid subnet mask.');

    // compute the subnet address using bit-wise AND
    sub_net1 := net1 And netmask1;
    sub_net2 := net2 And netmask2;
    sub_host1 := host1 And hostmask1;
    sub_host2 := host2 And hostmask2;

    // display the subnet address
    MaskEdit3.Text :=
      IntToStr(sub_net1) + '.' + IntToStr(sub_net2) + '.' +
      IntToStr(sub_host1) + '.' + IntToStr(sub_host2);
  end;
  {
  This example uses an edit box and a memo control on a form.
  When either the edit box or the memo is the active control,
  it is colored yellow. When the active control becomes
  inactive, the color of the control returns to the Windows
  system color for a window.  These event handlers could also
  be shared.
  }
  procedure TForm1.Edit1Enter(Sender: TObject);
  begin
    Edit1.Color := clYellow;
  end;

  procedure TForm1.Edit1Exit(Sender: TObject);
  begin
    Edit1.Color := clWindow;
  end;

  procedure TForm1.Memo1Enter(Sender: TObject);
  begin
    Memo1.Color := clYellow;
  end;

  procedure TForm1.Memo1Exit(Sender: TObject);
  begin
    Memo1.Color := clWindow;
  end;
```

See Also

OnEnter

ActiveControl

63.1.44.6.9 TRibbonSpinEdit.OnKeyDown Event

Occurs when a user presses any key while the control has focus.

Pascal

```
property OnKeyDown;
```

C++

```
__property OnKeyDown;
```

Description

Use the OnKeyDown event handler to specify special processing to occur when a key is pressed. The OnKeyDown handler can respond to keyboard keys, including function keys and keys combined with the `Shift`, `Alt`, and `Ctrl` keys, and pressed mouse buttons.

Note: TCheckBox does not fire OnKeyDown events for arrow keys.

The TKeyEvent type points to a method that handles keyboard events.

The Key parameter is the key on the keyboard. For non-alphanumeric keys, use virtual key codes to determine the key pressed. For more information, see Virtual Key codes.

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

An application gets Windows WM_KEYDOWN messages for all keys when the user presses a key. These messages indirectly fire the OnKeyDown event. If you set the Key to 0 there you prevent only any further processing of this message. But for keys that generate characters Windows also produces WM_CHAR. At the time your OnKeyDown event fires, the WM_CHAR message for the key will already be in the message queue. Setting Key to 0 does not stop it from being delivered, so it fires the OnKeyPress event. You must set the Key to #0 to prevent the control from intercepting the WM_CHAR message.

This method of organizing key processing has advantages. Code that only deals with characters, including control characters like #13 for carriage return, #3 for Ctrl-C and so on, should go into the OnKeyPress event. Code that deals with keys that do not generate characters should be put into the OnKeyDown event.

C++ Examples:

```
/*
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
*/
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  if (Key == VK_F1)
  {
    FormColor = Form1->Color;
    Form1->Color = clAqua;
    if (ComboBox1->Text == "KeyDown")
      ComboBox1->Text = "KeyDown2";
    else
      ComboBox1->Text = "KeyDown";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}

void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key, TShiftState Shift)
{
  if (Key == VK_F1)
  {
    Form1->Color = FormColor;
```

```
    if (ComboBox1->Text == "KeyUp")
      ComboBox1->Text = "KeyUp2";
    else
      ComboBox1->Text = "KeyUp";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}
/*
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
*/
#include <Printers.hpp>

void __fastcall TForm1::FormCreate(TObject *Sender)
{
   KeyPreview = True;

}

void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)

{
  if (Key == VK_ESCAPE && Printer()->Printing)
  {
    Printer()->Abort();
    MessageDlg("Printing aborted", mtInformation,
               TMsgDlgButtons() << mbOK,0);
  }
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TMemo *Memo1 = new TMemo(Form1); // The parent (Form1) will clean up this object.
  Memo1->Parent = Form1;
  Memo1->Visible = True;
  Memo1->Width = 700;
  if (OpenDialog1->Execute())
  {
    Memo1->Lines->LoadFromFile(OpenDialog1->FileName);

    Printer()->BeginDoc();
    int X = 200;
    int Y = 200;
    for (int I = 0; I <= 10; I++)
      if (!Printer()->Aborted)
      {
        Printer()->Canvas->TextOut(X, Y, Memo1->Lines->Strings[I]);
        Y = Y + 80;
        if (Y > (Printer()->PageHeight - 300))
        {
          Y = 200;
          Printer()->NewPage();
          Sleep(1000);  // to give you time to abort!
        }
      }
    if (!Printer()->Aborted) Printer()->EndDoc();
  }
}
```

Delphi Examples:

```
{
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
```

```
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
}
var
  FormColor: TColor;
procedure TForm1.FormCreate(Sender: TObject);
begin
  KeyPreview := True;
  FormColor := Form1.Color;
end;

procedure TForm1.FormKeyDown(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := clAqua;
    if (ComboBox1.Text = 'KeyDown') then
      ComboBox1.Text := 'KeyDown2'
    else
      ComboBox1.Text := 'KeyDown';
    WriteLn(ComboBox1.Text);
  end;
end;

procedure TForm1.FormKeyUp(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := FormColor;
    if (ComboBox1.Text = 'KeyUp') then
      ComboBox1.Text := 'KeyUp2'
    else
      ComboBox1.Text := 'KeyUp';
    WriteLn(ComboBox1.Text);
  end;
end;
{
The following code aborts a print job if the user presses
Esc. Note that you should set KeyPreview to true to ensure
that the OnKeyDown event handler of Form1 is called.
}

uses Printers;

procedure TForm1.Button1Click(Sender: TObject);
var
  I, X, Y: Integer;
  Memo1 : TMemo;
  r: TRect;
begin
  Memo1 := TMemo.Create(Form1);
  Memo1.Parent := Form1;
  Memo1.Visible := True;
  Memo1.Width := 700;
  if (OpenDialog1.Execute) then
  begin
    Memo1.Lines.LoadFromFile(OpenDialog1.FileName);

    Printer.BeginDoc;
    X := 200;
    Y := 200;
    for I := 0 to 140 do
```

```
        if (not Printer.Aborted) then
        begin
          Printer.Canvas.TextOut(X, Y, Memo1.Lines[I]);
          Y := Y + 80;
          if (Y > (Printer.PageHeight - 300)) then
          begin
            Y := 200;
            Printer.NewPage;
            Sleep(1000);  // to give you time to abort!
          end;
        end;
      if (not Printer.Aborted) then Printer.EndDoc;
    end;
  end;

  procedure TForm1.FormCreate(Sender: TObject);
  begin
     KeyPreview := True;
  end;

  procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
  begin
  if (Key=VK_ESCAPE) and Printer.Printing then
    begin
    Printer.Abort;
    MessageDlg('Printing aborted', mtInformation, [mbOK],0);
    end;
  end;
```

See Also

KeyPreview

OnKeyPress

OnKeyUp

63.1.44.6.10 TRibbonSpinEdit.OnKeyPress Event

Occurs when a key is pressed.

Pascal

```
property OnKeyPress;
```

C++

```
__property OnKeyPress;
```

Description

Use the OnKeyPress event handler to make something happen as a result of a single character key press.

The Key parameter in the OnKeyPress event handler is of type Char; therefore, the OnKeyPress event registers the ASCII character of the key pressed. Keys that don't correspond to an ASCII Char value (Shift or F1, for example) don't generate an OnKeyPress event. Key combinations (such as Shift+A), generate only one OnKeyPress event (for this example, Shift+A results in a Key value of "A" if Caps Lock is off). To respond to non-ASCII keys or key combinations, use the OnKeyDown or OnKeyUp event handlers.

An application gets Windows WM_KEYDOWN messages for all keys when the user presses a key. These messages indirectly fire the OnKeyDown event. If you set the Key to 0 there you prevent only any further processing of this message. But for keys that generate characters Windows also produces WM_CHAR. At the time your OnKeyDown event fires, the WM_CHAR message for the key will already be in the message queue. Setting Key to 0 does not stop it from being delivered, so it fires the OnKeyPress event. You must set the Key to #0 to prevent the control from intercepting the WM_CHAR message.

This method of organizing key processing has advantages. Code that only deals with characters, including control characters like #13 for carriage return, #3 for Ctrl-C and so on, should go into the OnKeyPress event. Code that deals with keys that do not generate characters should be put into the OnKeyDown event.

C++ Examples:

```
/*
This event handler displays a message dialog box specifying
which key was pressed:
*/

void __fastcall TForm1::FormKeyPress(TObject *Sender, char &Key)
{
  char keyString[25];
  keyString[0] = Key;
  strcpy(&keyString[1], " Was Pressed");
  TMsgDlgButtons() << mbOK;
  MessageDlg(AnsiString(keyString), TMsgDlgType(mtInformation), TMsgDlgButtons() << mbOK, 0);
}
```

Delphi Examples:

```
{
This event handler displays a message dialog box specifying
which key was pressed:
}
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  MessageDlg(Key + ' has been pressed', mtInformation, [mbOK], 0)
end;
```

See Also

OnKeyDown

OnKeyUp

KeyPreview

63.1.44.6.11 TRibbonSpinEdit.OnKeyUp Event

Occurs when the user releases a key that has been pressed.

Pascal

```
property OnKeyUp;
```

C++

```
__property OnKeyUp;
```

Description

Use the OnKeyUp event handler to provide special processing that occurs when a key is released. The OnKeyUp handler can respond to all keyboard keys, including function keys and keys combined with the Shift, Alt, and Ctrl keys.

The TKeyEvent type points to a method that handles keyboard events. The Key parameter is the key on the keyboard. For non-alphanumeric keys, you must use virtual key codes to determine the key pressed. For more information, see Virtual Key codes.

The Shift parameter indicates whether the Shift, Alt, or Ctrl keys are combined with the keystroke.

C++ Examples:

```
/*
This example changes a form’s color to aqua when the user
```

```
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
*/
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  if (Key == VK_F1)
  {
    FormColor = Form1->Color;
    Form1->Color = clAqua;
    if (ComboBox1->Text == "KeyDown")
      ComboBox1->Text = "KeyDown2";
    else
      ComboBox1->Text = "KeyDown";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}

void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key, TShiftState Shift)
{
  if (Key == VK_F1)
  {
    Form1->Color = FormColor;
    if (ComboBox1->Text == "KeyUp")
      ComboBox1->Text = "KeyUp2";
    else
      ComboBox1->Text = "KeyUp";
    Form1->Refresh();
    Memo1->Lines->Add(ComboBox1->Text);
  }
}
```

Delphi Examples:

```
{
This example changes a form's color to aqua when the user
presses the F1 key, even when a control on the form has the
focus. When the user releases the key, the form returns to
its original color.  Set the KeyPreview on the form to True.
Set the OnKeyDown event to FormKeyDown and set the OnKeyUp
event to FormKeyUp.  Set the KeyPreview on the form to True.
Behavior is dependent on keyboard driver(USB, serial, etc).
}
var
  FormColor: TColor;
procedure TForm1.FormCreate(Sender: TObject);
begin
  KeyPreview := True;
  FormColor := Form1.Color;
end;

procedure TForm1.FormKeyDown(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := clAqua;
    if (ComboBox1.Text = 'KeyDown') then
      ComboBox1.Text := 'KeyDown2'
    else
      ComboBox1.Text := 'KeyDown';
    WriteLn(ComboBox1.Text);
```

```
  end;
end;

procedure TForm1.FormKeyUp(
  Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if Key = VK_F1 then
  begin
    Form1.Color := FormColor;
    if (ComboBox1.Text = 'KeyUp') then
      ComboBox1.Text := 'KeyUp2'
    else
      ComboBox1.Text := 'KeyUp';
    WriteLn(ComboBox1.Text);
  end;
end;
```

See Also

OnKeyDown

OnKeyPress

63.1.44.6.12 TRibbonSpinEdit.OnMouseDown Event

Occurs when the user presses a mouse button with the mouse pointer over a control.

Pascal

```
property OnMouseDown;
```

C++

```
__property OnMouseDown;
```

Description

Use the OnMouseDown event handler to implement any special processing that should occur as a result of pressing a mouse button.

The OnMouseDown event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. X and Y are the pixel coordinates of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
```

```
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  Form1.Canvas.Rectangle(StartX, StartY, X, Y);
```

```
    StatusBar1.Panels[0].Text := '';
    StatusBar1.Panels[1].Text := '';
    StatusBar1.Panels[2].Text := '';
    StatusBar1.Panels[3].Text := '';
  end;

  procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
  begin
    if ssLeft in Shift then { make sure button is down }
    begin
      if Y > StartY then
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
        end
      else
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
        end;
      if X > StartX then
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
        end
      else
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
        end;
    end;
  end;
```

See Also

OnClick

OnMouseMove

OnMouseUp

MouseCapture

63.1.44.6.13 TRibbonSpinEdit.OnMouseMove Event

Occurs when the user moves the mouse pointer while the mouse pointer is over a control.

Pascal

```
property OnMouseMove;
```

C++

```
__property OnMouseMove;
```

Description

Use the OnMouseMove event handler to respond when the mouse pointer moves after the control has captured the mouse.

Use the Shift parameter of the OnMouseDown event handler, to determine to the state of the shift keys and mouse buttons. Shift keys are the Shift, Ctrl, and Alt keys or shift key-mouse button combinations. X and Y are pixel coordinates of the new location of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
```

```
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
```

```
  released, the rectangle appears on the form's canvas. Its
  top-left and bottom-right corners are defined by the
  location of the mouse pointer when the user pressed and
  released the mouse button.  While the user drags the mouse,
  the location of the top, left, bottom, and right sides of
  the rectangle are displayed in the status bar.
  }
  procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
  Y: Integer);
  begin
    StartX := X;
    StartY := Y;
  end;

  procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
  Integer);
  begin
    Form1.Canvas.Rectangle(StartX, StartY, X, Y);
    StatusBar1.Panels[0].Text := '';
    StatusBar1.Panels[1].Text := '';
    StatusBar1.Panels[2].Text := '';
    StatusBar1.Panels[3].Text := '';
  end;

  procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
  begin
    if ssLeft in Shift then { make sure button is down }
    begin
      if Y > StartY then
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
        end
      else
        begin
        StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
        StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
        end;
      if X > StartX then
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
        end
      else
        begin
        StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
        StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
        end;
    end;
  end;
```

See Also

OnMouseDown

OnMouseUp

MouseCapture

63.1.44.6.14 TRibbonSpinEdit.OnMouseUp Event

Occurs when the user releases a mouse button that was pressed with the mouse pointer over a component.

Pascal

```
  property OnMouseUp;
```

C++

```
__property OnMouseUp;
```

Description

Use an OnMouseUp event handler to implement special processing when the user releases a mouse button.

The OnMouseUp event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. X and Y are the pixel coordinates of the mouse pointer in the client area of the Sender.

C++ Examples:

```
/*
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
*/
int StartX, StartY; // Declare at the top of the form's unit
// Use this code as the OnMouseDown event handler of the form:
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  StartX = X;
  StartY = Y;
}

// Use this code as the OnMouseUp event handler of the form:
void __fastcall TForm1::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y)
{
  Form1->Canvas->Rectangle(StartX, StartY, X, Y);
  StatusBar1->Panels->Items[0]->Text = "";
  StatusBar1->Panels->Items[1]->Text = "";
  StatusBar1->Panels->Items[2]->Text = "";
  StatusBar1->Panels->Items[3]->Text = "";
}

// Use this code as the OnMouseMove event handler of the form:
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssLeft)) // make sure button is down
  {
    if (Y > StartY)
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(StartY);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(Y);
    }
    else
    {
      StatusBar1->Panels->Items[0]->Text = "Top: " + IntToStr(Y);
      StatusBar1->Panels->Items[2]->Text = "Bottom: " + IntToStr(StartY);
    }
    if (X > StartX)
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(StartX);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(X);
```

```
    }
    else
    {
      StatusBar1->Panels->Items[1]->Text = "Left: " + IntToStr(X);
      StatusBar1->Panels->Items[3]->Text = "Right: " + IntToStr(StartX);
    }
  }
}
```

Delphi Examples:

```
{
The following example requires a form with a four-paneled
status bar.  (Set the Width of the status panels to 150
before running this example).  When the user presses a mouse
button, moves the mouse, and releases the mouse button, a
rectangle is drawn on the form. When the mouse button is
released, the rectangle appears on the form's canvas. Its
top-left and bottom-right corners are defined by the
location of the mouse pointer when the user pressed and
released the mouse button.  While the user drags the mouse,
the location of the top, left, bottom, and right sides of
the rectangle are displayed in the status bar.
}
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
begin
  StartX := X;
  StartY := Y;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  Form1.Canvas.Rectangle(StartX, StartY, X, Y);
  StatusBar1.Panels[0].Text := '';
  StatusBar1.Panels[1].Text := '';
  StatusBar1.Panels[2].Text := '';
  StatusBar1.Panels[3].Text := '';
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if ssLeft in Shift then { make sure button is down }
  begin
    if Y > StartY then
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(StartY);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(Y);
      end
    else
      begin
      StatusBar1.Panels[0].Text := 'Top: ' + IntToStr(Y);
      StatusBar1.Panels[2].Text := 'Bottom: ' + IntToStr(StartY);
      end;
    if X > StartX then
      begin
      StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(StartX);
      StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(X);
      end
    else
      begin
      StatusBar1.Panels[1].Text := 'Left: ' + IntToStr(X);
      StatusBar1.Panels[3].Text := 'Right: ' + IntToStr(StartX);
      end;
  end;
end;
```

See Also

OnMouseDown

OnMouseMove

Responding to a Mouse-up Action

63.1.44.6.15 TRibbonSpinEdit.OnStartDrag Event

Occurs when the user begins to drag the control or an object it contains by left-clicking on the control and holding the mouse button down.

Pascal

```
property OnStartDrag;
```

C++

```
__property OnStartDrag;
```

Description

Use the OnStartDrag event handler to implement special processing when the user starts to drag the control or an object it contains. OnStartDrag only occurs if DragKind is dkDrag.

Sender is the control that is about to be dragged, or that contains the object about to be dragged.

The OnStartDrag event handler can create a TDragControlObjectEx instance for the DragObject parameter to specify the drag cursor, or, optionally, a drag image list. If you create a TDragControlObjectEx instance, there is no need to call the Free method for the DragObject when dragging is over. If you create, instead, a TDragControlObject instance, your application is responsible for freeing the drag object instance.

If the OnStartDrag event handler sets the DragObject parameter to nil (Delphi) or NULL (C++), a TDragControlObject object is automatically created and dragging begins on the control itself.

Note: On some controls, such as TRichEdit, the underlying Windows control handles internal drag operations. For these controls, there are no OnStartDrag or OnEndDrag events for drag operations within the control.

See Also

OnEndDrag

TDragControlObjectEx

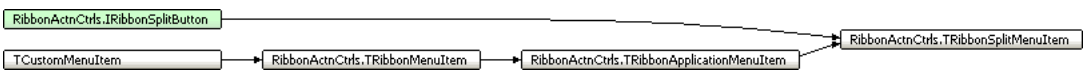
DragKind

OnDragOver

OnDragDrop

63.1.45 TRibbonSplitMenuItem Class

Class Hierarchy



Pascal

```
TRibbonSplitMenuItem = class(TRibbonApplicationMenuItem, IRibbonSplitButton);
```

C++

```
class TRibbonSplitMenuItem : public TRibbonApplicationMenuItem, public IRibbonSplitButton;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonSplitMenuItem.

63.1.45.1 TRibbonSplitMenuItem Members

The following tables list the members exposed by TRibbonSplitMenuItem.

Protected Fields

	Name	Description
	FMouseInSplit (see page 3898)	This is FMouseInSplit, a member of class TRibbonSplitMenuItem.
	FSplitRect (see page 3899)	This is FSplitRect, a member of class TRibbonSplitMenuItem.

Protected Methods

	Name	Description
	CalcLayout (see page 3757)	This is CalcLayout, a member of class TRibbonMenuItem.
	DrawBackground (see page 3900)	Draws a background image on the control.
	DrawGlyph (see page 3614)	Draws a symbol on the control.
	DrawGlyph (see page 3758)	Draws a symbol on the control.
	DrawSelectedBackground (see page 3900)	This is DrawSelectedBackground, a member of class TRibbonSplitMenuItem.
	DrawSeparator (see page 3615)	This is DrawSeparator, a member of class TRibbonApplicationMenuItem.
	DrawSeparator (see page 3759)	This is DrawSeparator, a member of class TRibbonMenuItem.
	DrawShadowedText (see page 3615)	This is DrawShadowedText, a member of class TRibbonApplicationMenuItem.
	DrawShadowedText (see page 3759)	This is DrawShadowedText, a member of class TRibbonMenuItem.
	DrawSubMenuGlyph (see page 3615)	This is DrawSubMenuGlyph, a member of class TRibbonApplicationMenuItem.
	DrawText (see page 3616)	Draws text on the control.
	DrawText (see page 3759)	Draws text on the control.
	GetImageSize (see page 3616)	This is GetImageSize, a member of class TRibbonApplicationMenuItem.
	GetShowShortCut (see page 3759)	This is GetShowShortCut, a member of class TRibbonMenuItem.
	IsInSplit (see page 3900)	This is IsInSplit, a member of class TRibbonSplitMenuItem.
	MouseMove (see page 3901)	Respond to mouse moving over control area..
	RibbonStyle (see page 3760)	This is RibbonStyle, a member of class TRibbonMenuItem.

Protected Properties

	Name	Description
	DrawBanner (see page 3760)	This is DrawBanner, a member of class TRibbonMenuItem.
	MouseInSplit (see page 3902)	This is MouseInSplit, a member of class TRibbonSplitMenuItem.

Public Methods

	Name	Description
	CalcBounds (see page 3899)	Calculates the size of the menu item.
	Click (see page 3899)	Executes the control's function.
	Create (see page 3898)	Creates an instance of TCustomMenuItem.
	UpdateSplitBounds (see page 3901)	This is UpdateSplitBounds, a member of class TRibbonSplitMenuItem.

63.1.45.2 TRibbonSplitMenuItem.Create Constructor

Creates an instance of TCustomMenuItem.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonSplitMenuItem(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TCustomMenuItem. However, you should never attempt to instantiate a TCustomMenuItem. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

After calling the inherited constructor, Create initializes the control, including the following property settings:

Initializes the TCanvas object for its Canvas property.

When overriding Create, always call the inherited Create method first, then proceed with the control's initialization. Remember to specify the override directive when overriding the Create method.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Create

Destroy

63.1.45.3 TRibbonSplitMenuItem Fields

The fields of the TRibbonSplitMenuItem class are listed here.

Protected Fields

	Name	Description
	FMouseInSplit (see page 3898)	This is FMouseInSplit, a member of class TRibbonSplitMenuItem.
	FSplitRect (see page 3899)	This is FSplitRect, a member of class TRibbonSplitMenuItem.

63.1.45.3.1 TRibbonSplitMenuItem.FMouseInSplit Field

Pascal

```
FMouseInSplit: Boolean;
```

C++

```
Boolean FMouseInSplit;
```

Description

This is FMouseInSplit, a member of class TRibbonSplitMenuItem.

63.1.45.3.2 TRibbonSplitMenuItem.FSplitRect Field

Pascal

```
FSplitRect: TRect;
```

C++

```
TRect FSplitRect;
```

Description

This is FSplitRect, a member of class TRibbonSplitMenuItem.

63.1.45.4 TRibbonSplitMenuItem Methods

The methods of the TRibbonSplitMenuItem class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3900)	Draws a background image on the control.
	DrawSelectedBackground (see page 3900)	This is DrawSelectedBackground, a member of class TRibbonSplitMenuItem.
	IsInSplit (see page 3900)	This is IsInSplit, a member of class TRibbonSplitMenuItem.
	MouseMove (see page 3901)	Respond to mouse moving over control area..

Public Methods

	Name	Description
	CalcBounds (see page 3899)	Calculates the size of the menu item.
	Click (see page 3899)	Executes the control's function.
	UpdateSplitBounds (see page 3901)	This is UpdateSplitBounds, a member of class TRibbonSplitMenuItem.

63.1.45.4.1 TRibbonSplitMenuItem.CalcBounds Method

Calculates the size of the menu item.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as whether or not the menu item is a separator or not. CalcBounds computes the boundaries of the menu item considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.45.4.2 TRibbonSplitMenuItem.Click Method

Executes the control's function.

Pascal

```
procedure Click; override;
```

C++

```
virtual __fastcall Click();
```

Description

This override of the method inherited from TCustomActionControl does little except bring the form containing the item into focus, and prevent the inherited method from being called. The reason for this lack of action is that at runtime, no action should be taken until the primary mouse button is pressed and released, which makes the OnClick event handler responsible for taking action. Click takes place only when the mouse button is first pressed, prior to release.

See Also

OnClick

63.1.45.4.3 TRibbonSplitMenuItem.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.45.4.4 TRibbonSplitMenuItem.DrawSelectedBackground Method

Pascal

```
procedure DrawSelectedBackground(ARect: TRect); override;
```

C++

```
virtual __fastcall DrawSelectedBackground(TRect ARect);
```

Description

This is DrawSelectedBackground, a member of class TRibbonSplitMenuItem.

63.1.45.4.5 TRibbonSplitMenuItem.IsInSplit Method

Pascal

```
function IsInSplit(const Pt: TPoint): Boolean;
```

C++

```
__fastcall Boolean IsInSplit(const TPoint Pt);
```

Description

This is IsInSplit, a member of class TRibbonSplitMenuItem.

63.1.45.4.6 TRibbonSplitMenuItem.MouseMove Method

Respond to mouse moving over control area..

Pascal

```
procedure MouseMove(Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseMove(TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseMove method to provide other responses in addition to calling the OnMouseMove event handler when the user moves the mouse.

A control calls MouseMove in response to any of the Windows mouse-move messages (WM_MOUSEMOVE), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively.

As the mouse cursor moves across a control, this method is called repeatedly. Each time it is called, it is with the new coordinates that reflect the continuous path of the mouse cursor across the screen real estate covered by the control's visual representation.

See Also

MouseDown

MouseUp

OnMouseMove

63.1.45.4.7 TRibbonSplitMenuItem.UpdateSplitBounds Method

Pascal

```
procedure UpdateSplitBounds; virtual;
```

C++

```
virtual __fastcall UpdateSplitBounds();
```

Description

This is UpdateSplitBounds, a member of class TRibbonSplitMenuItem.

63.1.45.5 TRibbonSplitMenuItem Properties

The properties of the TRibbonSplitMenuItem class are listed here.

Protected Properties

	Name	Description
	MouseInSplit (see page 3902)	This is MouseInSplit, a member of class TRibbonSplitMenuItem.

63.1.45.5.1 TRibbonSplitMenuItem.MouseInSplit Property

Pascal

```
property MouseInSplit: Boolean;
```

C++

```
__property Boolean MouseInSplit;
```

Description

This is MouseInSplit, a member of class TRibbonSplitMenuItem.

63.1.46 TRibbonStyleMenuButton Class

Top level menus when using Ribbon (see page 3347) Styles in applications not using Ribbon (see page 3347) interface.

Class Hierarchy

Pascal

```
TRibbonStyleMenuButton = class(TCustomMenuButton);
```

C++

```
class TRibbonStyleMenuButton : public TCustomMenuButton;
```

File

RibbonActnCtrls

Description

TRibbonStyleMenuButton is used for the top level menus when using one of the Ribbon (see page 3347) Styles in an application that doesn't use the Ribbon (see page 3347) interface.

63.1.46.1 TRibbonStyleMenuButton Members

The following tables list the members exposed by TRibbonStyleMenuButton.

Protected Methods

	Name	Description
	DrawBackground (see page 3903)	Draws a background image on the control.
	RibbonStyle (see page 3903)	This is RibbonStyle, a member of class TRibbonStyleMenuButton.

63.1.46.2 TRibbonStyleMenuButton Methods

The methods of the TRibbonStyleMenuButton class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3903)	Draws a background image on the control.
	RibbonStyle (see page 3903)	This is RibbonStyle, a member of class TRibbonStyleMenuButton.

63.1.46.2.1 TRibbonStyleMenuButton.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.46.2.2 TRibbonStyleMenuButton.RibbonStyle Method

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

This is RibbonStyle, a member of class TRibbonStyleMenuButton.

63.1.47 TRibbonTextControl Class

Class Hierarchy

Pascal

```
TRibbonTextControl = class(TCustomActionControl);
```

C++

```
class TRibbonTextControl : public TCustomActionControl;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonTextControl.

63.1.47.1 TRibbonTextControl Members

The following tables list the members exposed by TRibbonTextControl.

Protected Methods

	Name	Description
	DrawBackground (see page 3904)	Draws a background image on the control.
	DrawText (see page 3905)	Draws text on the control.

Public Methods

	Name	Description
	CalcBounds (see page 3904)	Calculates the size of the control.

63.1.47.2 TRibbonTextControl Methods

The methods of the TRibbonTextControl class are listed here.

Protected Methods

	Name	Description
	DrawBackground (see page 3904)	Draws a background image on the control.
	DrawText (see page 3905)	Draws text on the control.

Public Methods

	Name	Description
	CalcBounds (see page 3904)	Calculates the size of the control.

63.1.47.2.1 TRibbonTextControl.CalcBounds Method

Calculates the size of the control.

Pascal

```
procedure CalcBounds; override;
```

C++

```
virtual __fastcall CalcBounds();
```

Description

The size of the control depends on many factors, such as the presence of an icon, caption, and shortcuts. CalcBounds computes the boundaries of the control considering these kinds of factors. At the end, it calls SetBounds and other routines to ensure that the values are set properly.

See Also

SetBounds

63.1.47.2.2 TRibbonTextControl.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.47.2.3 TRibbonTextControl.DrawText Method

Draws text on the control.

Pascal

```
procedure DrawText(var ARect: TRect; var Flags: Cardinal; Text: string); override;
```

C++

```
virtual __fastcall DrawText(TRect ARect, unsigned Flags, UnicodeString Text);
```

Description

Use DrawText to draw the text specified by Text on the control within the area specified by ARect.

See Also

DrawBackground

DrawGlyph

63.1.48 TRibbonToolScrollButton Class

Class Hierarchy

Pascal

```
TRibbonToolScrollButton = class(TCustomToolScrollBtn);
```

C++

```
class TRibbonToolScrollButton : public TCustomToolScrollBtn;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TRibbonToolScrollButton.

63.1.48.1 TRibbonToolScrollButton Members

The following tables list the members exposed by TRibbonToolScrollButton.

Protected Methods

	Name	Description
	DrawArrows (see page 3906)	This is DrawArrows, a member of class TRibbonToolScrollButton.
	DrawBackground (see page 3906)	Draws a background image on the control.

63.1.48.2 TRibbonToolScrollButton Methods

The methods of the TRibbonToolScrollButton class are listed here.

Protected Methods

	Name	Description
	DrawArrows (see page 3906)	This is DrawArrows, a member of class TRibbonToolScrollButton.
	DrawBackground (see page 3906)	Draws a background image on the control.

63.1.48.2.1 TRibbonToolScrollButton.DrawArrows Method

Pascal

```
procedure DrawArrows; override;
```

C++

```
virtual __fastcall DrawArrows();
```

Description

This is DrawArrows, a member of class TRibbonToolScrollButton.

63.1.48.2.2 TRibbonToolScrollButton.DrawBackground Method

Draws a background image on the control.

Pascal

```
procedure DrawBackground(var PaintRect: TRect); override;
```

C++

```
virtual __fastcall DrawBackground(TRect PaintRect);
```

Description

Use DrawBackground to draw a background image on the control within the area specified by PaintRect.

See Also

DrawGlyph

DrawText

63.1.49 TTimerSpeedButton Class

Class Hierarchy

Pascal

```
TTimerSpeedButton = class(TSpeedButton);
```

C++

```
class TTimerSpeedButton : public TSpeedButton;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.TTimerSpeedButton.

63.1.49.1 TTimerSpeedButton Members

The following tables list the members exposed by TTimerSpeedButton.

Protected Methods

	Name	Description
	CMMouseenter (see page 3908)	This is CMMouseenter, a member of class TTimerSpeedButton.
	CMMouseleave (see page 3908)	This is CMMouseleave, a member of class TTimerSpeedButton.
	MouseDown (see page 3909)	OnMouseDown event dispatcher.
	MouseUp (see page 3909)	OnMouseUp event dispatcher.
	Paint (see page 3910)	Render control surface.

Protected Properties

	Name	Description
	MouseInControl (see page 3910)	This is MouseInControl, a member of class TTimerSpeedButton.

Public Methods

	Name	Description
	Create (see page 3907)	Creates and initializes an instance of TSpeedButton.
	Destroy (see page 3908)	Destroys an instance of TSpeedButton.

Public Properties

	Name	Description
	ButtonStyle (see page 3910)	This is ButtonStyle, a member of class TTimerSpeedButton.
	TimeBtnState (see page 3911)	This is TimeBtnState, a member of class TTimerSpeedButton.

63.1.49.2 TTimerSpeedButton.Create Constructor

Creates and initializes an instance of TSpeedButton.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TTimerSpeedButton(TComponent * AOwner);
```

Description

Call Create to create a speed button at runtime. Speed buttons placed on forms at design time are created automatically.

AOwner is the component, typically a form, that is responsible for freeing the speed button.

See Also

Owner

63.1.49.3 TTimerSpeedButton.Destroy Destructor

Destroys an instance of TSpeedButton.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TTimerSpeedButton();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the speed button is not nil, and only then calls Destroy.

See Also

System::TObject::Free

63.1.49.4 TTimerSpeedButton Methods

The methods of the TTimerSpeedButton class are listed here.

Protected Methods

	Name	Description
	CMMouseenter (see page 3908)	This is CMMouseenter, a member of class TTimerSpeedButton.
	CMMouseleave (see page 3908)	This is CMMouseleave, a member of class TTimerSpeedButton.
V	MouseDown (see page 3909)	OnMouseDown event dispatcher.
V	MouseUp (see page 3909)	OnMouseUp event dispatcher.
V	Paint (see page 3910)	Render control surface.

63.1.49.4.1 TTimerSpeedButton.CMMouseenter Method

Pascal

```
procedure CMMouseenter(var Message: TMessage); message CM_MOUSEENTER;
```

C++

```
__fastcall CMMouseenter(TMessage Message);
```

Description

This is CMMouseenter, a member of class TTimerSpeedButton.

63.1.49.4.2 TTimerSpeedButton.CMMouseleave Method

Pascal

```
procedure CMMouseleave(var Message: TMessage); message CM_MOUSELEAVE;
```

C++

```
__fastcall CMMouseleave(TMessage Message);
```

Description

This is CMMouseleave, a member of class TTimerSpeedButton.

63.1.49.4.3 TTimerSpeedButton.MouseDown Method

OnMouseDown event dispatcher.

Pascal

```
procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer);
override;
```

C++

```
virtual __fastcall MouseDown(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user presses the mouse button down while the cursor's hotspot is over the control.

The Button parameter determines which mouse button the user pressed. Shift indicates which shift keys (Shift, Ctrl, or Alt) were down when the user pressed the mouse button. X and Y are the pixel coordinates of the mouse pointer within the client area of the control.

A control calls MouseDown in response to any of the Windows mouse-down messages (WM_LBUTTONDOWN, WM_MBUTTONDOWN, WM_RBUTTONDOWN), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was pressed down: left, right, or middle.

See Also

MouseMove

MouseUp

OnMouseDown

63.1.49.4.4 TTimerSpeedButton.MouseUp Method

OnMouseUp event dispatcher.

Pascal

```
procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X: Integer; Y: Integer); override;
```

C++

```
virtual __fastcall MouseUp(TMouseButton Button, TShiftState Shift, int X, int Y);
```

Description

Override the protected MouseDown method to provide other responses in addition to calling the OnMouseDown event handler when the user releases a previously pressed mouse button while the cursor's hotspot is over the control.

A control calls MouseUp in response to any of the Windows mouse-up messages (WM_LBUTTONUP, WM_MBUTTONUP, WM_RBUTTONUP), decoding the message parameters into the shift-key state and position, which it passes in the Shift, X, and Y parameters, respectively. The value of the Button parameter indicates which mouse button was released: left, right, or middle

See Also

MouseDown

MouseMove

OnMouseUp

63.1.49.4.5 TTimerSpeedButton.Paint Method

Render control surface.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method is called automatically when a lightweight control needs update its display area. Code in the Paint method uses the Canvas property to render the control surface.

As implemented in TGraphicControl, Paint does nothing. Specific rendering logic is provided by Paint methods in descendant control classes.

See Also

Canvas

63.1.49.5 TTimerSpeedButton Properties

The properties of the TTimerSpeedButton class are listed here.

Protected Properties

	Name	Description
	MouseInControl (see page 3910)	This is MouseInControl, a member of class TTimerSpeedButton.

Public Properties

	Name	Description
	ButtonStyle (see page 3910)	This is ButtonStyle, a member of class TTimerSpeedButton.
	TimeBtnState (see page 3911)	This is TimeBtnState, a member of class TTimerSpeedButton.

63.1.49.5.1 TTimerSpeedButton.ButtonStyle Property

Pascal

```
property ButtonStyle: TSpinButtonStyle;
```

C++

```
__property TSpinButtonStyle ButtonStyle;
```

Description

This is ButtonStyle, a member of class TTimerSpeedButton.

63.1.49.5.2 TTimerSpeedButton.MouseInControl Property

Pascal

```
property MouseInControl: Boolean;
```

C++

```
__property Boolean MouseInControl;
```

Description

This is MouseInControl, a member of class TTimerSpeedButton.

63.1.49.5.3 TTimerSpeedButton.TimeBtnState Property

Pascal

```
property TimeBtnState: TTimeBtnState;
```

C++

```
__property TTimeBtnState TimeBtnState;
```

Description

This is TimeBtnState, a member of class TTimerSpeedButton.

63.2 Interfaces

The following table lists interfaces in this documentation.

Interfaces

	Name	Description
	IRibbonBarKeyTip (see page 3912)	Action Bars that support KeyTips should implement this interface.
	IRibbonContainedControl (see page 3914)	Interface implemented by controls used as containers for displaying other controls.
	IRibbonControl (see page 3916)	Interface implemented by controls visible in Ribbon (see page 3347) Groups.
	IRibbonGroupDropdown (see page 3917)	This is class RibbonActnCtrls.IRibbonGroupDropdown.
	IRibbonKeyTip (see page 3918)	Commands with a **KeyTip (see page 3922)** should implement this interface.
	IRibbonSplitButton (see page 3922)	Interface implemented by controls that have split button.

63.2.1 IRibbonBarKeyTip Interface

Action Bars that support KeyTips should implement this interface.

Class Hierarchy

RibbonActnCtrls.IRibbonBarKeyTip

Pascal

```
IRibbonBarKeyTip = interface;
```

C++

```
__interface IRibbonBarKeyTip;
```

File

RibbonActnCtrls

Description

Action Bars that support KeyTips should implement IRibbonBarKeyTip. The TRibbonQuickAccessToolbar and TRibbonGroup classes implement this interface.

See Also

TRibbonGroup (see page 3500)

TRibbonQuickAccessToolbar (see page 3546)

63.2.1.1 IRibbonBarKeyTip Members

The following tables list the members exposed by IRibbonBarKeyTip.

Public Methods

	Name	Description
	BarHandleKeyTips (see page 3913)	Check if any commands on Action Bar are partial or full match for **KeyTip**.
	BarHideKeyTips (see page 3913)	Hide any visible **KeyTip** windows.

	BarShowKeyTips (see page 3914)	Show **KeyTip**s for any commands that support them.

63.2.1.2 IRibbonBarKeyTip Methods

The methods of the IRibbonBarKeyTip class are listed here.

Public Methods

	Name	Description
	BarHandleKeyTips (see page 3913)	Check if any commands on Action Bar are partial or full match for **KeyTip**.
	BarHideKeyTips (see page 3913)	Hide any visible **KeyTip** windows.
	BarShowKeyTips (see page 3914)	Show **KeyTip**s for any commands that support them.

63.2.1.2.1 IRibbonBarKeyTip.BarHandleKeyTips Method

Check if any commands on Action Bar are partial or full match for **KeyTip**.

Pascal

```
procedure BarHandleKeyTips(const KeyTip: string; var KeyTips: TKeyTipList);
```

C++

```
__fastcall BarHandleKeyTips(const UnicodeString KeyTip, TKeyTipList KeyTips);
```

Description

BarHandleKeyTips checks if any commands on the Action Bar are a partial or full match for the **KeyTip**. Matches are returned as a TKeyTipMatchItem (see page 3598) in the `List` parameter. If only one item is a match, that command is executed.

See Also

TKeyTipMatchItem (see page 3598)

BarHideKeyTips (see page 3913)

BarShowKeyTips (see page 3914)

63.2.1.2.2 IRibbonBarKeyTip.BarHideKeyTips Method

Hide any visible **KeyTip** windows.

Pascal

```
procedure BarHideKeyTips(KeyTips: TKeyTipList);
```

C++

```
__fastcall BarHideKeyTips(TKeyTipList KeyTips);
```

Description

BarHideKeyTips hides any visible **KeyTip** windows. `ExceptList` may contain a list of TKeyTipMatchItem (see page 3598) objects for commands that should not be hidden. These items are current partial matches and require additional handling.

See Also

TKeyTipMatchItem (see page 3598)

BarHandleKeyTips (see page 3913)

BarShowKeyTips (see page 3914)

63.2.1.2.3 IRibbonBarKeyTip.BarShowKeyTips Method

Show **KeyTip**s for any commands that support them.

Pascal

```
procedure BarShowKeyTips;
```

C++

```
__fastcall BarShowKeyTips();
```

Description

BarShowKeyTips shows **KeyTip**s for any commands that support them.

See Also

BarHandleKeyTips (see page 3913)

BarHideKeyTips (see page 3913)

63.2.2 IRibbonContainedControl Interface

Interface implemented by controls used as containers for displaying other controls.

Class Hierarchy

Pascal

```
IRibbonContainedControl = interface(IRibbonControl);
```

C++

```
__interface IRibbonContainedControl : public IRibbonControl;
```

File

RibbonActnCtrls

Description

IRibbonContainedControl is an interface implemented by a control that is used as a container for displaying other controls.

See Also

IRibbonControl (see page 3916)

IRibbonSplitButton (see page 3922)

63.2.2.1 IRibbonContainedControl Members

The following tables list the members exposed by IRibbonContainedControl.

Public Methods

	Name	Description
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.
	GetContainedControl (see page 3915)	This is GetContainedControl, a member of class IRibbonContainedControl.
	SetContainedControl (see page 3915)	This is SetContainedControl, a member of class IRibbonContainedControl.

Public Properties

	Name	Description
	ContainedControl (see page 3915)	This is ContainedControl, a member of class IRibbonContainedControl.

63.2.2.2 IRibbonContainedControl Methods

The methods of the IRibbonContainedControl class are listed here.

Public Methods

	Name	Description
	GetContainedControl (see page 3915)	This is GetContainedControl, a member of class IRibbonContainedControl.
	SetContainedControl (see page 3915)	This is SetContainedControl, a member of class IRibbonContainedControl.

63.2.2.2.1 IRibbonContainedControl.GetContainedControl Method

Pascal

```
function GetContainedControl: TControl;
```

C++

```
__fastcall TControl * GetContainedControl();
```

Description

This is GetContainedControl, a member of class IRibbonContainedControl.

63.2.2.2.2 IRibbonContainedControl.SetContainedControl Method

Pascal

```
procedure SetContainedControl(const Value: TControl);
```

C++

```
__fastcall SetContainedControl(const TControl * Value);
```

Description

This is SetContainedControl, a member of class IRibbonContainedControl.

63.2.2.3 IRibbonContainedControl Properties

The properties of the IRibbonContainedControl class are listed here.

Public Properties

	Name	Description
	ContainedControl (see page 3915)	This is ContainedControl, a member of class IRibbonContainedControl.

63.2.2.3.1 IRibbonContainedControl.ContainedControl Property

Pascal

```
property ContainedControl: TControl;
```

C++

```
__property TControl * ContainedControl;
```

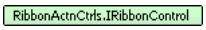
Description

This is ContainedControl, a member of class IRibbonContainedControl.

63.2.3 IRibbonControl Interface

Interface implemented by controls visible in Ribbon (see page 3347) Groups.

Class Hierarchy



Pascal

```
IRibbonControl = interface;
```

C++

```
__interface IRibbonControl;
```

File

RibbonActnCtrls

Description

IRibbonControl is an interface implemented by a control that is visible in Ribbon (see page 3347) Groups.

See Also

IRibbonContainedControl (see page 3914)

IRibbonSplitButton (see page 3922)

63.2.3.1 IRibbonControl Members

The following tables list the members exposed by IRibbonControl.

Public Methods

	Name	Description
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.

63.2.3.2 IRibbonControl Methods

The methods of the IRibbonControl class are listed here.

Public Methods

	Name	Description
	CanAddToQuickAccessToolbar (see page 3916)	This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.

63.2.3.2.1 IRibbonControl.CanAddToQuickAccessToolbar Method

Pascal

```
function CanAddToQuickAccessToolbar: Boolean;
```

C++

```
__fastcall Boolean CanAddToQuickAccessToolbar();
```

Description

This is CanAddToQuickAccessToolbar, a member of class IRibbonControl.

63.2.4 IRibbonGroupDropdown Interface

Class Hierarchy

RibbonActnCtrls.IRibbonGroupDropdown

Pascal

```
IRibbonGroupDropdown = interface;
```

C++

```
__interface IRibbonGroupDropdown;
```

File

RibbonActnCtrls

Description

This is class RibbonActnCtrls.IRibbonGroupDropdown.

63.2.4.1 IRibbonGroupDropdown Members

The following tables list the members exposed by IRibbonGroupDropdown.

Public Methods

	Name	Description
	CloseUp (see page 3917)	This is CloseUp, a member of class IRibbonGroupDropdown.
	GetParentControl (see page 3918)	This is GetParentControl, a member of class IRibbonGroupDropdown.
	GetTopPosition (see page 3918)	This is GetTopPosition, a member of class IRibbonGroupDropdown.

63.2.4.2 IRibbonGroupDropdown Methods

The methods of the IRibbonGroupDropdown class are listed here.

Public Methods

	Name	Description
	CloseUp (see page 3917)	This is CloseUp, a member of class IRibbonGroupDropdown.
	GetParentControl (see page 3918)	This is GetParentControl, a member of class IRibbonGroupDropdown.
	GetTopPosition (see page 3918)	This is GetTopPosition, a member of class IRibbonGroupDropdown.

63.2.4.2.1 IRibbonGroupDropdown.CloseUp Method

Pascal

```
procedure CloseUp;
```

C++

```
__fastcall CloseUp();
```

Description

This is CloseUp, a member of class IRibbonGroupDropdown.

63.2.4.2.2 IRibbonGroupDropdown.GetParentControl Method

Pascal

```
function GetParentControl: TWinControl;
```

C++

```
__fastcall TWinControl * GetParentControl();
```

Description

This is GetParentControl, a member of class IRibbonGroupDropdown.

63.2.4.2.3 IRibbonGroupDropdown.GetTopPosition Method

Pascal

```
function GetTopPosition: Integer;
```

C++

```
__fastcall int GetTopPosition();
```

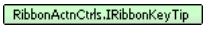
Description

This is GetTopPosition, a member of class IRibbonGroupDropdown.

63.2.5 IRibbonKeyTip Interface

Commands with a **KeyTip (see page 3922)** should implement this interface.

Class Hierarchy



Pascal

```
IRibbonKeyTip = interface;
```

C++

```
__interface IRibbonKeyTip;
```

File

RibbonActnCtrls

Description

Commands that have a **KeyTip (see page 3922)** should implement IRibbonKeyTip.

63.2.5.1 IRibbonKeyTip Members

The following tables list the members exposed by IRibbonKeyTip.

Public Methods

	Name	Description
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.2.5.2 IRibbonKeyTip Methods

The methods of the IRibbonKeyTip class are listed here.

Public Methods

	Name	Description
	GetKeyTip (see page 3919)	Returns current **KeyTip (see page 3922)** for command.
	HandleKeyTip (see page 3920)	Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).
	HideKeyTip (see page 3920)	Hide **KeyTip (see page 3922)** for command.
	KeyTipClick (see page 3920)	Handle clicking a **KeyTip (see page 3922)**.
	SetKeyTip (see page 3921)	Sets **KeyTip (see page 3922)** for command.
	ShowKeyTip (see page 3921)	Shows **KeyTip (see page 3922)** for command in specified position.

63.2.5.2.1 IRibbonKeyTip.GetKeyTip Method

Returns current **KeyTip (see page 3922)** for command.

Pascal

```
function GetKeyTip: string;
```

C++

```
__fastcall UnicodeString GetKeyTip();
```

Description

GetKeyTip returns the current **KeyTip (see page 3922)** for the command

See Also

KeyTip (see page 3922)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.2.5.2.2 IRibbonKeyTip.HandleKeyTip Method

Indicates if command's **KeyTip (see page 3922)** matches currently input KeyTip (see page 3922).

Pascal

```
procedure HandleKeyTip(const KeyTip: string; var Handled: TKeyTipHandled);
```

C++

```
__fastcall HandleKeyTip(const UnicodeString KeyTip, TKeyTipHandled Handled);
```

Description

HandleKeyTip indicates whether this command's

KeyTip (see page 3922) matches the currently input KeyTip (see page 3922) by the user. Full and partial matches are taken into consideration.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.2.5.2.3 IRibbonKeyTip.HideKeyTip Method

Hide **KeyTip (see page 3922)** for command.

Pascal

```
procedure HideKeyTip(const Match: string);
```

C++

```
__fastcall HideKeyTip(const UnicodeString Match);
```

Description

This method hides the **KeyTip (see page 3922)** for this command. If the **Match** parameter matches (full or partial) the command's **KeyTip (see page 3922)**, the **KeyTip (see page 3922)** remains visible.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

SetKeyTip (see page 3921)

ShowKeyTip (see page 3921)

63.2.5.2.4 IRibbonKeyTip.KeyTipClick Method

Handle clicking a **KeyTip (see page 3922)**.

Pascal

```
procedure KeyTipClick;
```

C++

```
__fastcall KeyTipClick();
```

Description

KeyTipClick Handle clicking a **KeyTip (see page 3922)**. Some requires need to be handled differently when clicked by a **KeyTip (see page 3922)**.

63.2.5.2.5 IRibbonKeyTip.SetKeyTip Method

Sets **KeyTip (see page 3922)** for command.

Pascal

```
procedure SetKeyTip(const Value: string);
```

C++

```
__fastcall SetKeyTip(const UnicodeString Value);
```

Description

This method sets the **KeyTip (see page 3922)** for the command.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

ShowKeyTip (see page 3921)

63.2.5.2.6 IRibbonKeyTip.ShowKeyTip Method

Shows **KeyTip (see page 3922)** for command in specified position.

Pascal

```
procedure ShowKeyTip(const Position: TKeyTipPosition);
```

C++

```
__fastcall ShowKeyTip(const TKeyTipPosition Position);
```

Description

This method shows the **KeyTip (see page 3922)** for the command in the position specified. Only the Application Button uses a **KeyTip (see page 3922)** position.

See Also

KeyTip (see page 3922)

GetKeyTip (see page 3919)

HandleKeyTip (see page 3920)

HideKeyTip (see page 3920)

SetKeyTip (see page 3921)

63.2.5.3 IRibbonKeyTip Properties

The properties of the IRibbonKeyTip class are listed here.

Public Properties

	Name	Description
	KeyTip (see page 3922)	Current **KeyTip** for command

63.2.5.3.1 IRibbonKeyTip.KeyTip Property

Current **KeyTip** for command

Pascal

```
property KeyTip: string;
```

C++

```
__property UnicodeString KeyTip;
```

Description

KeyTip is the current **KeyTip** for the command

See Also

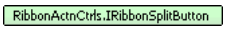
GetKeyTip (see page 3919)

SetKeyTip (see page 3921)

63.2.6 IRibbonSplitButton Interface

Interface implemented by controls that have split button.

Class Hierarchy



Pascal

```
IRibbonSplitButton = interface;
```

C++

```
__interface IRibbonSplitButton;
```

File

RibbonActnCtrls

Description

IRibbonSplitButton is an interface implemented by controls that have a split button.

See Also

IRibbonContainedControl (see page 3914)

IRibbonControl (see page 3916)

63.2.6.1 IRibbonSplitButton Members

The following tables list the members exposed by IRibbonSplitButton.

Public Methods

	Name	Description
	UpdateSplitBounds (see page 3923)	This is UpdateSplitBounds, a member of class IRibbonSplitButton.

63.2.6.2 IRibbonSplitButton Methods

The methods of the IRibbonSplitButton class are listed here.

Public Methods

	Name	Description
	UpdateSplitBounds (see page 3923)	This is UpdateSplitBounds, a member of class IRibbonSplitButton.

63.2.6.2.1 IRibbonSplitButton.UpdateSplitBounds Method

Pascal

```
procedure UpdateSplitBounds;
```

C++

```
__fastcall UpdateSplitBounds();
```

Description

This is UpdateSplitBounds, a member of class IRibbonSplitButton.

63.3 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	DrawLine (see page 3924)	This is function RibbonActnCtrls.DrawLine.
	IsLeftMouseButtonPressed (see page 3924)	This is function RibbonActnCtrls.IsLeftMouseButtonPressed.
	ShowKeyTip (see page 3924)	Show **KeyTip** for control.
	ShowKeyTip (see page 3924)	Show **KeyTip** for control.

63.3.1 RibbonActnCtrls.DrawLine Function

Pascal

```
procedure DrawLine(ACanvas: TCanvas; const AFromPt: TPoint; const AToPt: TPoint);
```

C++

```
DrawLine(TCanvas ACanvas, const TPoint AFromPt, const TPoint AToPt);
```

File

RibbonActnCtrls

Description

This is function RibbonActnCtrls.DrawLine.

63.3.2 RibbonActnCtrls.IsLeftMouseButtonPressed Function

Pascal

```
function IsLeftMouseButtonPressed(Control: TControl): Boolean;
```

C++

```
Boolean IsLeftMouseButtonPressed(TControl * Control);
```

File

RibbonActnCtrls

Description

This is function RibbonActnCtrls.IsLeftMouseButtonPressed.

63.3.3 RibbonActnCtrls.ShowKeyTip Function

Show **KeyTip** for control.

Pascal

```
function ShowKeyTip(const KeyTip: string; const Enabled: Boolean; const Control: TControl;
const Position: TKeyTipPosition): TCustomControl; overload;
function ShowKeyTip(const KeyTip: string; const Enabled: Boolean; const Center: Boolean; const
Rect: TRect; const Position: TKeyTipPosition): TCustomControl; overload;
```

C++

```
TCustomControl ShowKeyTip(const UnicodeString KeyTip, const Boolean Enabled, const TControl *
Control, const TKeyTipPosition Position);
TCustomControl ShowKeyTip(const UnicodeString KeyTip, const Boolean Enabled, const Boolean
Center, const TRect Rect, const TKeyTipPosition Position);
```

File

RibbonActnCtrls

Description

ShowKeyTip shows the **KeyTip** for control.

The parameters are:

- **KeyTip**: The **KeyTip** to display
- **Enabled**: If the **KeyTip** should have an enabled appearance
- **Control**: The control that the **KeyTip** is for. The **KeyTip** displays close to the control
- **Position** - The position of the **KeyTip**. The position of the **KeyTip** can be either **ktpCenter** or **ktpBottom**. Most **KeyTip**s are displayed with the bottom third of the control covered with the **KeyTip**.

63.4 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TKeyTipHandled (see page 3926)	Validate if command's **KeyTip** matches current user input.
	TKeyTipMatchType (see page 3927)	Type of match command's **KeyTip** is for current **KeyTip** value.
	TKeyTipPosition (see page 3927)	Position of KeyTips.
	TKeyTipState (see page 3928)	Current state of KeyTips.
	TQATScrollMode (see page 3929)	This is record RibbonActnCtrls.TQATScrollMode.
	TSpinButtonStyle (see page 3930)	This is record RibbonActnCtrls.TSpinButtonStyle.

63.4.1 RibbonActnCtrls.TKeyTipHandled Enumeration

Validate if command's **KeyTip** matches current user input.

Pascal

```
TKeyTipHandled = (
  ktNone,
  ktPartial,
  ktFull
);
```

C++

```
enum TKeyTipHandled {
  ktNone,
  ktPartial,
  ktFull
};
```

File

RibbonActnCtrls

Description

Use TKeyTipHandled to validate if a command's **KeyTip** matches the current user input. It can have one of the following values:

Value	Meaning
ktNone	Current KeyTip didn't match the command's **KeyTip**.
ktPartial	Current **KeyTip** matches the first character of the commands **KeyTip**.
ktFull	Current **KeyTip** is a full match for the command. The source **KeyTip** could be either one character or two characters.

See Also

TKeyTipMatchType (see page 3927)

TKeyTipPosition (see page 3927)

TKeyTipState (see page 3928)

63.4.2 RibbonActnCtrls.TKeyTipMatchType Enumeration

Type of match command's **KeyTip** is for current **KeyTip** value.

Pascal

```
TKeyTipMatchType = (
  mtApplicationMenu,
  mtTabs,
  mtControl,
  mtGroup,
  mtCollapsed,
  mtDialogButton
);
```

C++

```
enum TKeyTipMatchType {
  mtApplicationMenu,
  mtTabs,
  mtControl,
  mtGroup,
  mtCollapsed,
  mtDialogButton
};
```

File

RibbonActnCtrls

Description

TKeyTipMatchType is the type of match the command's **KeyTip** is for the current **KeyTip** value. It can have one of the following values:

Value	Meaning
mtApplicationMenu	For ApplicationMenu.
mtTabs	For Tabs.
mtControl	For a Control.
mtGroup	For a Group.
mtCollapsed	For a Collapsed item.
mtDialogButton	For a Dialog Button.

See Also

TKeyTipHandled (see page 3926)

TKeyTipPosition (see page 3927)

TKeyTipState (see page 3928)

63.4.3 RibbonActnCtrls.TKeyTipPosition Enumeration

Position of KeyTips.

Pascal

```
TKeyTipPosition = (
  ktpApplicationButton,
  ktpQuickAccessToolbar,
  ktpTabs,
  ktpGroupA,
  ktpGroupB,
  ktpGroupC,
  ktpGroupD
);
```

C++

```
enum TKeyTipPosition {
  ktpApplicationButton,
  ktpQuickAccessToolbar,
  ktpTabs,
  ktpGroupA,
  ktpGroupB,
  ktpGroupC,
  ktpGroupD
};
```

File

RibbonActnCtrls

Description

TKeyTipPosition is the position of KeyTips. It can have one of the following values:

Value	Meaning
ktpApplicationButton	On Application Button.
ktpQuickAccessToolbar	On Quick Access Toolbar.
ktpTabs	On tabs.
ktpGroupA	On Group A
ktpGroupB	On Group B
ktpGroupC	On Group C
ktpGroupD	On Group D

See Also

TKeyTipHandled (see page 3926)

TKeyTipMatchType (see page 3927)

TKeyTipState (see page 3928)

63.4.4 RibbonActnCtrls.TKeyTipState Enumeration

Current state of KeyTips.

Pascal

```
TKeyTipState = (
  ktsTabs,
  ktsControls,
```

```
    ktsCollapsedGroup
  );
```

C++

```
  enum TKeyTipState {
    ktsTabs,
    ktsControls,
    ktsCollapsedGroup
  };
```

File

RibbonActnCtrls

Description

TKeyTipState is the current state KeyTips are in. It can have one of the following values:

Value	Meaning
ktsTabs	Initial state where the user hasn't yet selected a Tab to display the `KeyTips` for. In this state, `KeyTips` are displayed for the Application Button, Quick Access Toolbar and the Tabs.
ktsControls	`KeyTips` are displayed for all commands on the Active Tab. `KeyTips` are also displayed for Groups dialog buttons and collapsed group buttons.
ktsCollapsedGroup	KeyTips are being displayed for commands that are on a collapsed group.

See Also

TKeyTipHandled (see page 3926)

TKeyTipMatchType (see page 3927)

TKeyTipPosition (see page 3927)

63.4.5 RibbonActnCtrls.TQATScrollMode Enumeration

Pascal

```
  TQATScrollMode = (
    qsCustomize,
    qsMoreControls
  );
```

C++

```
  enum TQATScrollMode {
    qsCustomize,
    qsMoreControls
  };
```

File

RibbonActnCtrls

Description

This is record RibbonActnCtrls.TQATScrollMode.

63.4.6 RibbonActnCtrls.TSpinButtonStyle Enumeration

Pascal

```
TSpinButtonStyle = (
  sbsUp,
  sbsDown
);
```

C++

```
enum TSpinButtonStyle {
  sbsUp,
  sbsDown
};
```

File

RibbonActnCtrls

Description

This is record RibbonActnCtrls.TSpinButtonStyle.

63.5 Types

The following table lists types in this documentation.

Types

Name	Description
TNumGlyphs (see page 3931)	This is type RibbonActnCtrls.TNumGlyphs.
TTimeBtnState (see page 3931)	This is type RibbonActnCtrls.TTimeBtnState.

63.5.1 RibbonActnCtrls.TNumGlyphs Type

Pascal

```
TNumGlyphs = Buttons.TNumGlyphs;
```

C++

```
Buttons.TNumGlyphs TNumGlyphs;
```

File

RibbonActnCtrls

Description

This is type RibbonActnCtrls.TNumGlyphs.

63.5.2 RibbonActnCtrls.TTimeBtnState Type

Pascal

```
TTimeBtnState = set of (tbFocusRect, tbAllowTimer);
```

C++

```
(tbFocusRect, tbAllowTimer) TTimeBtnState;
```

File

RibbonActnCtrls

Description

This is type RibbonActnCtrls.TTimeBtnState.

63.6 Constants

The following table lists constants in this documentation.

Constants

Name	Description
UM_FADECONTROL (see page 3932)	This is constant RibbonActnCtrls.UM_FADECONTROL.

63.6.1 RibbonActnCtrls.UM_FADECONTROL Constant

Pascal

```
UM_FADECONTROL = WM_USER + 100;
```

C++

```
UM_FADECONTROL = WM_USER + 100;
```

File

RibbonActnCtrls

Description

This is constant RibbonActnCtrls.UM_FADECONTROL.

64 RibbonActnMenus Namespace

This is namespace RibbonActnMenus.

Topics

Name	Description
Classes (see page 3934)	The following table lists classes in this documentation.

Classes

	Name	Description
	TOptionItem (see page 3934)	Item added to OptionItems collection for TRibbonApplicationMenuBar (see page 3942).
	TOptionItems (see page 3938)	Stores additional option items appearing in border of Application Menu.
	TRibbonActionPopupMenu (see page 3940)	Display top level menus when using Ribbon (see page 3347) Style without Ribbon (see page 3347) interface.
	TRibbonApplicationMenuBar (see page 3942)	Display Application Menu for Ribbon (see page 3347).
	TRibbonApplicationPopupMenu (see page 3961)	Ddisplay popup menu when Application Button clicked.
	TRibbonQATCustomizeMenu (see page 3968)	This is class RibbonActnMenus.TRibbonQATCustomizeMenu.
	TRibbonRecentDocuments (see page 3970)	This is class RibbonActnMenus.TRibbonRecentDocuments.
	TRibbonStyleMainPopupMenu (see page 3975)	This is class RibbonActnMenus.TRibbonStyleMainPopupMenu.
	TRibbonStylePopupMenu (see page 3977)	This is class RibbonActnMenus.TRibbonStylePopupMenu.

64.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TOptionItem (see page 3934)	Item added to OptionItems collection for TRibbonApplicationMenuBar (see page 3942).
	TOptionItems (see page 3938)	Stores additional option items appearing in border of Application Menu.
	TRibbonActionPopupMenu (see page 3940)	Display top level menus when using Ribbon (see page 3347) Style without Ribbon (see page 3347) interface.
	TRibbonApplicationMenuBar (see page 3942)	Display Application Menu for Ribbon (see page 3347).
	TRibbonApplicationPopupMenu (see page 3961)	Ddisplay popup menu when Application Button clicked.
	TRibbonQATCustomizeMenu (see page 3968)	This is class RibbonActnMenus.TRibbonQATCustomizeMenu.
	TRibbonRecentDocuments (see page 3970)	This is class RibbonActnMenus.TRibbonRecentDocuments.
	TRibbonStyleMainPopupMenu (see page 3975)	This is class RibbonActnMenus.TRibbonStyleMainPopupMenu.
	TRibbonStylePopupMenu (see page 3977)	This is class RibbonActnMenus.TRibbonStylePopupMenu.

64.1.1 TOptionItem Class

Item added to OptionItems collection for TRibbonApplicationMenuBar (see page 3942).

Class Hierarchy

Pascal

```
TOptionItem = class(TCollectionItem);
```

C++

```
class TOptionItem : public TCollectionItem;
```

File

RibbonActnMenus

Description

TOptionItem is an item added to the OptionItems collection for the TRibbonApplicationMenuBar (see page 3942) component. This class represents the action to use for the command as well as the **KeyTip (see page 3937)** to use.

See Also

TRibbonApplicationMenuBar (see page 3942)

OptionItems (see page 3956)

64.1.1.1 TOptionItem Members

The following tables list the members exposed by TOptionItem.

Protected Methods

	Name	Description
	GetDisplayName (see page 3936)	Returns the name of the collection item as it appears in the collection editor.

Public Methods

	Name	Description
	Assign (see page 3935)	Copies the contents of another, similar object.

Published Properties

	Name	Description
	Action (see page 3936)	This is Action, a member of class TOptionItem.
	Caption (see page 3937)	This is Caption, a member of class TOptionItem.
	Hint (see page 3937)	This is Hint, a member of class TOptionItem.
	KeyTip (see page 3937)	This is KeyTip, a member of class TOptionItem.
	Tag (see page 3937)	This is Tag, a member of class TOptionItem.

64.1.1.2 TOptionItem Methods

The methods of the TOptionItem class are listed here.

Protected Methods

	Name	Description
	GetDisplayName (see page 3936)	Returns the name of the collection item as it appears in the collection editor.

Public Methods

	Name	Description
	Assign (see page 3935)	Copies the contents of another, similar object.

64.1.1.2.1 TOptionItem.Assign Method

Copies the contents of another, similar object.

Pascal

```
procedure Assign(Source: TPersistent); override;
```

C++

```
virtual __fastcall Assign(TPersistent * Source);
```

Description

Call Assign to copy the properties or other attributes of one object from another. The standard form of a call to Assign is

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

which tells the Destination object to copy the contents of the Source object to itself.

Most objects override Assign to handle the assignment of properties from similar objects. When overriding Assign, call the inherited method if the destination object can't handle the assignment of properties from the class of the Source parameter.

If no overridden Assign method can handle the assignment of properties from Source, the method implemented in TPersistent calls the source object's AssignTo method. This allows the source object to handle the assignment. If the Source object is nil (Delphi) or NULL (C++), Assign raises an EConvertError exception.

In general, the statement

```
Destination := Source; {Delphi}
```

```
Destination = Source; // C++
```

is not the same as the statement

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

The assignment operator makes Destination reference the same object as Source, whereas the Assign method copies the contents of the object referenced by Source into the object referenced by Destination.

Note: The types of some properties are also objects. If these properties have write methods that use Assign to set the value of the property, then in these cases the assignment operator does the same thing as the Assign method.

See Also

AssignTo

EConvertError

64.1.1.2.2 TOptionItem.GetDisplayName Method

Returns the name of the collection item as it appears in the collection editor.

Pascal

```
function GetDisplayName: string; override;
```

C++

```
virtual __fastcall UnicodeString GetDisplayName();
```

Description

GetDisplayName is the protected read implementation of the DisplayName property.

See Also

DisplayName

GetNamePath

64.1.1.3 TOptionItem Properties

The properties of the TOptionItem class are listed here.

Published Properties

	Name	Description
	Action (see page 3936)	This is Action, a member of class TOptionItem.
	Caption (see page 3937)	This is Caption, a member of class TOptionItem.
	Hint (see page 3937)	This is Hint, a member of class TOptionItem.
	KeyTip (see page 3937)	This is KeyTip, a member of class TOptionItem.
	Tag (see page 3937)	This is Tag, a member of class TOptionItem.

64.1.1.3.1 TOptionItem.Action Property

Pascal

```
property Action: TContainedAction;
```

C++

```
__property TContainedAction Action;
```

Description

This is Action, a member of class TOptionItem.

64.1.1.3.2 TOptionItem.Caption Property

Pascal

```
property Caption: string;
```

C++

```
__property UnicodeString Caption;
```

Description

This is Caption, a member of class TOptionItem.

64.1.1.3.3 TOptionItem.Hint Property

Pascal

```
property Hint: string;
```

C++

```
__property UnicodeString Hint;
```

Description

This is Hint, a member of class TOptionItem.

64.1.1.3.4 TOptionItem.KeyTip Property

Pascal

```
property KeyTip: string;
```

C++

```
__property UnicodeString KeyTip;
```

Description

This is KeyTip, a member of class TOptionItem.

64.1.1.3.5 TOptionItem.Tag Property

Pascal

```
property Tag: Integer;
```

C++

```
__property int Tag;
```

Description

This is Tag, a member of class TOptionItem.

64.1.2 TOptionItems Class

Stores additional option items appearing in border of Application Menu.

Class Hierarchy

Pascal

```
TOptionItems = class(TOwnedCollection);
```

C++

```
class TOptionItems : public TOwnedCollection;
```

File

RibbonActnMenus

Description

TOptionItems stores the additional option items that appear in the border of the Application Menu.

See Also

TRibbonApplicationMenuBar (see page 3942)

64.1.2.1 TOptionItems Members

The following tables list the members exposed by TOptionItems.

Public Methods

	Name	Description
	Add (see page 3938)	Creates a new TCollectionItem instance and adds it to the Items array.
	Insert (see page 3939)	Creates a new TCollectionItem instance and adds it to the Items array.

Public Properties

	Name	Description
R	ApplicationMenu (see page 3939)	This is ApplicationMenu, a member of class TOptionItems.
	Items (see page 3940)	Lists the items in the collection.

64.1.2.2 TOptionItems Methods

The methods of the TOptionItems class are listed here.

Public Methods

	Name	Description
	Add (see page 3938)	Creates a new TCollectionItem instance and adds it to the Items array.
	Insert (see page 3939)	Creates a new TCollectionItem instance and adds it to the Items array.

64.1.2.2.1 TOptionItems.Add Method

Creates a new TCollectionItem instance and adds it to the Items array.

Pascal

```
function Add: TOptionItem;
```

C++

```
__fastcall TOptionItem Add();
```

Description

Call Add to create an item in the collection. The new item is placed at the end of the Items array.

Add returns the new collection item.

See Also

Items

Insert

Create

Delete

64.1.2.2.2 TOptionItems.Insert Method

Creates a new TCollectionItem instance and adds it to the Items array.

Pascal

```
function Insert(Index: Integer): TOptionItem;
```

C++

```
__fastcall TOptionItem Insert(int Index);
```

Description

Call Insert to add a new item at a specified position in the collection. Existing items (starting from the specified position) are moved up in the Items array.

Insert returns the new collection item.

See Also

Add

Items

64.1.2.3 TOptionItems Properties

The properties of the TOptionItems class are listed here.

Public Properties

	Name	Description
	ApplicationMenu (see page 3939)	This is ApplicationMenu, a member of class TOptionItems.
	Items (see page 3940)	Lists the items in the collection.

64.1.2.3.1 TOptionItems.ApplicationMenu Property

Pascal

```
property ApplicationMenu: TRibbonApplicationMenuBar;
```

C++

```
__property TRibbonApplicationMenuBar ApplicationMenu;
```

Description

This is ApplicationMenu, a member of class TOptionItems.

64.1.2.3.2 TOptionItems.Items Property

Lists the items in the collection.

Pascal

```
property Items [Index: Integer]: TOptionItem;
```

C++

```
__property TOptionItem Items[int Index];
```

Description

Use Items to access individual items in the collection. The value of the `Index` parameter corresponds to the `Index` property of TCollectionItem. It represents the position of the item in the collection.

See Also

Count

ItemClass

Index

Add

Insert

64.1.3 TRibbonActionPopupMenu Class

Display top level menus when using Ribbon (see page 3347) Style without Ribbon (see page 3347) interface.

Class Hierarchy

Pascal

```
TRibbonActionPopupMenu = class(TXPStylePopupMenu);
```

C++

```
class TRibbonActionPopupMenu : public TXPStylePopupMenu;
```

File

RibbonActnMenus

Description

TRibbonActionPopupMenu displays top level menus when using the Ribbon (see page 3347) Style without the Ribbon (see page 3347) interface.

64.1.3.1 TRibbonActionPopupMenu Members

The following tables list the members exposed by TRibbonActionPopupMenu.

Protected Methods

	Name	Description
	NCPaint (see page 3941)	This is NCPaint, a member of class TRibbonActionPopupMenu.

Public Methods

	Name	Description
	CreateControl (see page 3941)	This is CreateControl, a member of class TRibbonActionPopupMenu.
	TrackMenu (see page 3942)	Executes the action associated with the menu item.

Public Properties

	Name	Description
	SelectFirstItem (see page 3942)	This is SelectFirstItem, a member of class TRibbonActionPopupMenu.

64.1.3.2 TRibbonActionPopupMenu Methods

The methods of the TRibbonActionPopupMenu class are listed here.

Protected Methods

	Name	Description
	NCPaint (see page 3941)	This is NCPaint, a member of class TRibbonActionPopupMenu.

Public Methods

	Name	Description
	CreateControl (see page 3941)	This is CreateControl, a member of class TRibbonActionPopupMenu.
	TrackMenu (see page 3942)	Executes the action associated with the menu item.

64.1.3.2.1 TRibbonActionPopupMenu.CreateControl Method

Pascal

```
function CreateControl(AnItem: TActionClientItem): TCustomActionControl; override;
```

C++

```
virtual __fastcall TCustomActionControl CreateControl(TActionClientItem AnItem);
```

Description

This is CreateControl, a member of class TRibbonActionPopupMenu.

64.1.3.2.2 TRibbonActionPopupMenu.NCPaint Method

Pascal

```
procedure NCPaint(DC: HDC); override;
```

C++

```
virtual __fastcall NCPaint(HDC DC);
```

Description

This is NCPaint, a member of class TRibbonActionPopupMenu.

64.1.3.2.3 TRibbonActionPopupMenu.TrackMenu Method

Executes the action associated with the menu item.

Pascal

```
procedure TrackMenu; override;
```

C++

```
virtual __fastcall TrackMenu();
```

Description

TrackMenu is called internally to execute the action associated with the selected menu item..

64.1.3.3 TRibbonActionPopupMenu Properties

The properties of the TRibbonActionPopupMenu class are listed here.

Public Properties

	Name	Description
	SelectFirstItem (see page 3942)	This is SelectFirstItem, a member of class TRibbonActionPopupMenu.

64.1.3.3.1 TRibbonActionPopupMenu.SelectFirstItem Property

Pascal

```
property SelectFirstItem: Boolean;
```

C++

```
__property Boolean SelectFirstItem;
```

Description

This is SelectFirstItem, a member of class TRibbonActionPopupMenu.

64.1.4 TRibbonApplicationMenuBar Class

Display Application Menu for Ribbon (see page 3347).

Class Hierarchy

Pascal

```
TRibbonApplicationMenuBar = class(TCustomActionMainMenuBar);
```

C++

```
class TRibbonApplicationMenuBar : public TCustomActionMainMenuBar;
```

File

RibbonActnMenus

Description

TRibbonApplicationMenuBar is used to display the Application Menu for the Ribbon (see page 3347).

The TRibbonApplicationMenuBar component is not available on the component palette and should not be created dynamically. To add an Application Menu to your Ribbon (see page 3347), select the Add Application Menu command in the Ribbon (see page 3347) component editor.

The Application Menu consists of two vertical panes. On the left is a list of Menu items. You can modify the menu items at design time.

On the right is a place to display Recent items. This area is mostly intended for use as a recently used file list. To use this as a Recent Items pane, use the TRibbon.AddRecentItem method to add items. There are a number of Recent Item functions to help you setup the list of items. You can also use the right pane to add more commands. Do this by setting the TApplicationMenu.CommandType to **ctCommand**. See the CommandType property for more details.

The bottom section of the Application button can also display commands. Usually this section is set aside for the Application Exit command and Application Options command. Set your commands in the OptionItems (see page 3956) collection.

To modify the contents of the Application Menu, you can drag actions from the customize dialog and drop them in the menu on the left of the application menu.

See Also

TApplicationMenu (see page 3350)

TRibbon (see page 3480)

CommandType (see page 3354)

AddRecentItem (see page 3370)

64.1.4.1 TRibbonApplicationMenuBar Members

The following tables list the members exposed by TRibbonApplicationMenuBar.

Protected Methods

	Name	Description
	CreateControls (see page 3946)	This is CreateControls, a member of class TRibbonApplicationMenuBar.
	DragOver (see page 3946)	OnDragOver event dispatcher.
	DrawBackground (see page 3946)	This is DrawBackground, a member of class TRibbonApplicationMenuBar.
	GetControlClass (see page 3946)	This is GetControlClass, a member of class TRibbonApplicationMenuBar.
	NewPopup (see page 3947)	This is NewPopup, a member of class TRibbonApplicationMenuBar.
	Notification (see page 3947)	Responds to notifications that components are being created or destroyed.
	Paint (see page 3948)	This is Paint, a member of class TRibbonApplicationMenuBar.
	RibbonStyle (see page 3948)	This is RibbonStyle, a member of class TRibbonApplicationMenuBar.
	ShowKeyTips (see page 3949)	This is ShowKeyTips, a member of class TRibbonApplicationMenuBar.
	WMSysCommand (see page 3949)	This is WMSysCommand, a member of class TRibbonApplicationMenuBar.

Public Methods

	Name	Description
	CloseMenu (see page 3945)	Closes the menu currently being displayed.
	Create (see page 3944)	Creates and initializes a TCustomActionMainMenuBar instance.
	CreateControl (see page 3945)	This is CreateControl, a member of class TRibbonApplicationMenuBar.
	Destroy (see page 3944)	Destroys an instance of TCustomActionMainMenuBar.
	GetRibbon (see page 3947)	This is GetRibbon, a member of class TRibbonApplicationMenuBar.
	SelectApplicationButton (see page 3948)	This is SelectApplicationButton, a member of class TRibbonApplicationMenuBar.
	SetBounds (see page 3948)	Sets the bounds based on whether the popup is expanded or not
	SetRibbon (see page 3949)	This is SetRibbon, a member of class TRibbonApplicationMenuBar.

Public Properties

	Name	Description
	Align (see page 3950)	Determines how the control aligns within its container (parent control).
	HorzMargin (see page 3955)	Specifies the width of the horizontal margins in pixels.
	Spacing (see page 3956)	Determines where the image and text appear on an action band's rendered control that represents the action client.
	VertMargin (see page 3958)	Specifies the width of the vertical margins in pixels.
R	VisibleMenu (see page 3959)	This is VisibleMenu, a member of class TRibbonApplicationMenuBar.
R	VisibleMenuCount (see page 3959)	This is VisibleMenuCount, a member of class TRibbonApplicationMenuBar.

Published Properties

	Name	Description
	ActionManager (see page 3950)	Specifies the action manager component that is associated with the action band.
	Height (see page 3954)	Specifies the vertical size of the control in pixels.
	Left (see page 3955)	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
	OptionItems (see page 3956)	Commands displayed at bottom of Application Menu.
	RecentItems (see page 3956)	Recent items displayed in right pane of Application.
	Top (see page 3957)	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
	Width (see page 3959)	Specifies the horizontal size of the control or form in pixels.

64.1.4.2 TRibbonApplicationMenuBar.Create Constructor

Creates and initializes a TCustomActionMainMenuBar instance.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonApplicationMenuBar(TComponent * AOwner);
```

Description

Create generates a TCustomActionMainMenuBar and initializes its properties.

```
__fastcall virtual TCustomActionMainMenuBar (Classes::TComponentTComponent_Object* AOwner);
__fastcall TCustomActionMainMenuBar (HWND ParentWindow) :
Controls::TWinControlTWinControl_Create(ParentWindow) { }
```

64.1.4.3 TRibbonApplicationMenuBar.Destroy Destructor

Destroys an instance of TCustomActionMainMenuBar.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRibbonApplicationMenuBar();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TCustomActionMainMenuBar reference is not nil, and only then calls Destroy.

See Also

System::TObject::Free

Destroy

64.1.4.4 TRibbonApplicationMenuBar Methods

The methods of the TRibbonApplicationMenuBar class are listed here.

Protected Methods

	Name	Description
	CreateControls (see page 3946)	This is CreateControls, a member of class TRibbonApplicationMenuBar.
	DragOver (see page 3946)	OnDragOver event dispatcher.
	DrawBackground (see page 3946)	This is DrawBackground, a member of class TRibbonApplicationMenuBar.
	GetControlClass (see page 3946)	This is GetControlClass, a member of class TRibbonApplicationMenuBar.
	NewPopup (see page 3947)	This is NewPopup, a member of class TRibbonApplicationMenuBar.
	Notification (see page 3947)	Responds to notifications that components are being created or destroyed.
	Paint (see page 3948)	This is Paint, a member of class TRibbonApplicationMenuBar.
	RibbonStyle (see page 3948)	This is RibbonStyle, a member of class TRibbonApplicationMenuBar.
	ShowKeyTips (see page 3949)	This is ShowKeyTips, a member of class TRibbonApplicationMenuBar.
	WMSysCommand (see page 3949)	This is WMSysCommand, a member of class TRibbonApplicationMenuBar.

Public Methods

	Name	Description
	CloseMenu (see page 3945)	Closes the menu currently being displayed.
	CreateControl (see page 3945)	This is CreateControl, a member of class TRibbonApplicationMenuBar.
	GetRibbon (see page 3947)	This is GetRibbon, a member of class TRibbonApplicationMenuBar.
	SelectApplicationButton (see page 3948)	This is SelectApplicationButton, a member of class TRibbonApplicationMenuBar.
	SetBounds (see page 3948)	Sets the bounds based on whether the popup is expanded or not
	SetRibbon (see page 3949)	This is SetRibbon, a member of class TRibbonApplicationMenuBar.

64.1.4.4.1 TRibbonApplicationMenuBar.CloseMenu Method

Closes the menu currently being displayed.

Pascal

```
procedure CloseMenu; override;
```

C++

```
virtual __fastcall CloseMenu();
```

Description

CloseMenu is called to close the menu currently being displayed, including any submenus that may be open.

64.1.4.4.2 TRibbonApplicationMenuBar.CreateControl Method

Pascal

```
function CreateControl(AnItem: TActionClientItem): TCustomActionControl; override;
```

C++

```
virtual __fastcall TCustomActionControl CreateControl(TActionClientItem AnItem);
```

Description

This is CreateControl, a member of class TRibbonApplicationMenuBar.

64.1.4.4.3 TRibbonApplicationMenuBar.CreateControls Method

Pascal

```
procedure CreateControls; override;
```

C++

```
virtual __fastcall CreateControls();
```

Description

This is CreateControls, a member of class TRibbonApplicationMenuBar.

64.1.4.4.4 TRibbonApplicationMenuBar.DragOver Method

OnDragOver event dispatcher.

Pascal

```
procedure DragOver(Source: TObject; X: Integer; Y: Integer; State: TDragState; var Accept:
Boolean); override;
```

C++

```
virtual __fastcall DragOver(TObject * Source, int X, int Y, TDragState State, Boolean Accept);
```

Description

Override DragOver to add additional code that executes before the OnDragOver event handler is called.

DragOver sets the Accept parameter to true to indicate that the user can drop the dragged object on the control. It sets Accept to false to indicate that the user cannot drop the dragged object on the control.

The Source parameter is the object being dragged. The State parameter indicates how the dragged object is moving in relation to the control. X and Y indicate the current position of the mouse.

See Also

DragDrop

OnDragOver

64.1.4.4.5 TRibbonApplicationMenuBar.DrawBackground Method

Pascal

```
procedure DrawBackground; override;
```

C++

```
virtual __fastcall DrawBackground();
```

Description

This is DrawBackground, a member of class TRibbonApplicationMenuBar.

64.1.4.4.6 TRibbonApplicationMenuBar.GetControlClass Method

Pascal

```
function GetControlClass(AnItem: TActionClientItem): TCustomActionControlClass; override;
```

C++

```
virtual __fastcall TCustomActionControlClass GetControlClass(TActionClientItem AnItem);
```

Description

This is GetControlClass, a member of class TRibbonApplicationMenuBar.

64.1.4.4.7 TRibbonApplicationMenuBar.GetRibbon Method

Pascal

```
function GetRibbon: TWinControl;
```

C++

```
__fastcall TWinControl * GetRibbon();
```

Description

This is GetRibbon, a member of class TRibbonApplicationMenuBar.

64.1.4.4.8 TRibbonApplicationMenuBar.NewPopup Method

Pascal

```
function NewPopup: TCustomActionPopupMenu; override;
```

C++

```
virtual __fastcall TCustomActionPopupMenu NewPopup();
```

Description

This is NewPopup, a member of class TRibbonApplicationMenuBar.

64.1.4.4.9 TRibbonApplicationMenuBar.Notification Method

Responds to notifications that components are being created or destroyed.

Pascal

```
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
```

C++

```
virtual __fastcall Notification(TComponent * AComponent, TOperation Operation);
```

Description

Notification allows TControl to update its PopupMenu property when the pop-up menu it refers to is destroyed.

Override the Notification method to respond to notifications that other components are about to be destroyed or have just been created. Use the Notification method to update controls that rely on other objects. Remove references to objects that are going away so that the control does not refer to objects that have been destroyed. For example, data-aware objects override the Notification method to update themselves when their data source is removed. Some objects respond to notifications that other objects of a specific type are being created. For example, the session component sets the Session property of new data-aware controls to itself when the AutoSessionName property is true.

By default, components pass along the notification to their owned components, if any.

See Also

PopupMenu

Notification

64.1.4.4.10 TRibbonApplicationMenuBar.Paint Method

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

This is Paint, a member of class TRibbonApplicationMenuBar.

64.1.4.4.11 TRibbonApplicationMenuBar.RibbonStyle Method

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

This is RibbonStyle, a member of class TRibbonApplicationMenuBar.

64.1.4.4.12 TRibbonApplicationMenuBar.SelectApplicationButton Method

Pascal

```
procedure SelectApplicationButton;
```

C++

```
__fastcall SelectApplicationButton();
```

Description

This is SelectApplicationButton, a member of class TRibbonApplicationMenuBar.

64.1.4.4.13 TRibbonApplicationMenuBar.SetBounds Method

Sets the bounds based on whether the popup is expanded or not

Pascal

```
procedure SetBounds(ALeft: Integer; ATop: Integer; AWidth: Integer; AHeight: Integer);
override;
```

C++

```
virtual __fastcall SetBounds(int ALeft, int ATop, int AWidth, int AHeight);
```

Description

SetBounds calls the inherited method after altering the AWidth and AHeight parameters (if necessary) to include the width and height of the expanded popup menu. For more information about SetBounds, see the documentation for the inherited method.

See Also

SetBounds

64.1.4.4.14 TRibbonApplicationMenuBar.SetRibbon Method

Pascal

```
procedure SetRibbon(const Value: TWinControl);
```

C++

```
__fastcall SetRibbon(const TWinControl * Value);
```

Description

This is SetRibbon, a member of class TRibbonApplicationMenuBar.

64.1.4.4.15 TRibbonApplicationMenuBar.ShowKeyTips Method

Pascal

```
procedure ShowKeyTips;
```

C++

```
__fastcall ShowKeyTips();
```

Description

This is ShowKeyTips, a member of class TRibbonApplicationMenuBar.

64.1.4.4.16 TRibbonApplicationMenuBar.WMSysCommand Method

Pascal

```
procedure WMSysCommand(var Message: TWMSysCommand); message WM_SYSCOMMAND;
```

C++

```
__fastcall WMSysCommand(TWMSysCommand Message);
```

Description

This is WMSysCommand, a member of class TRibbonApplicationMenuBar.

64.1.4.5 TRibbonApplicationMenuBar Properties

The properties of the TRibbonApplicationMenuBar class are listed here.

Public Properties

	Name	Description
	Align (see page 3950)	Determines how the control aligns within its container (parent control).
	HorzMargin (see page 3955)	Specifies the width of the horizontal margins in pixels.
	Spacing (see page 3956)	Determines where the image and text appear on an action band's rendered control that represents the action client.
	VertMargin (see page 3958)	Specifies the width of the vertical margins in pixels.
R	VisibleMenu (see page 3959)	This is VisibleMenu, a member of class TRibbonApplicationMenuBar.
R	VisibleMenuCount (see page 3959)	This is VisibleMenuCount, a member of class TRibbonApplicationMenuBar.

Published Properties

	Name	Description
	ActionManager (see page 3950)	Specifies the action manager component that is associated with the action band.
	Height (see page 3954)	Specifies the vertical size of the control in pixels.

	Left (see page 3955)	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
	OptionItems (see page 3956)	Commands displayed at bottom of Application Menu.
	RecentItems (see page 3956)	Recent items displayed in right pane of Application.
	Top (see page 3957)	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
	Width (see page 3959)	Specifies the horizontal size of the control or form in pixels.

64.1.4.5.1 TRibbonApplicationMenuBar.ActionManager Property

Specifies the action manager component that is associated with the action band.

Pascal

```
property ActionManager;
```

C++

```
__property ActionManager;
```

Description

Use ActionManager to access the action manager that uses this action band to render UI elements it manages. Typically, this association is created when you drag actions from the action manager and drop them on the action bar. When an action is dropped on an action band and the ActionManager property is not assigned, the ActionManager property is automatically assigned to the action manager that manages the action that was dropped on the action band.

At runtime, you can perform the equivalent operation to dragging an action onto the action band by setting the ActionBar property of one of the action clients that the action manager's ActionBars property lists so that it specifies this component.

See Also

ActionBars

ActionClient

ActionBar

64.1.4.5.2 TRibbonApplicationMenuBar.Align Property

Determines how the control aligns within its container (parent control).

Pascal

```
property Align;
```

C++

```
__property Align;
```

Description

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes. When the parent is resized, an aligned control also resizes so that it continues to span the top, bottom, left, or right edge of the parent.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel.

Tip: If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

Note: To cause a control to maintain a specified relationship with an edge of its parent, but not necessarily lie along one edge of the parent, use the Anchors property instead.

C++ Examples:

```
/*
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
*/

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  TSplitter* ps = new TSplitter(Form1); // These objects will all be cleaned up by the owner
(Form1).
  TFileListBox* pflb = new TFileListBox(Form1); // The parent (Form1) will clean up these
objects.
  TDirectoryListBox* pdlb = new TDirectoryListBox(Form1);

  // Line up the directory list box on the left of the form.
  pdlb->Parent = Form1;
  pdlb->Align = alLeft;
  pdlb->Width = Form1->ClientWidth/3;

  // Now use the splitter to divide the directory pane from the file pane.
  ps->Parent = Form1;
  ps->Left = pdlb->Left + pdlb->Width + 1; // Move to right of directory list.
  ps->Align = pdlb->Align; // Give same alignment as directory list.

  // Each pane must be at least one quarter of the form's width.
  ps->MinSize = Form1->ClientWidth/4;

  // Finally, create the last pane - a file list box.
  pflb->Parent = Form1;
  pflb->Align = alClient;
  pdlb->FileList = pflb; // Link the file list box to the directory list box.
}
/*
Dynamically create a TProgressBar control and align it to
the bottom of the form.
*/

#include <ComCtrls.hpp>
#include <Controls.hpp>

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TProgressBar *ProgressBar = new TProgressBar(this);  // The owner will clean this up.
  ProgressBar->Parent = this;
  ProgressBar->Align = alBottom;
  TButton *Button1 = new TButton(this);  // The owner will clean this up.
  Button1->Parent = this;
}
/*
This example dynamically creates a Page Control, then a
```

```
series of Tab Sheets on the Page Control.
*/
#include <Comctrls.hpp>
TPageControl* ppc;
const int MAXTABS = 3;
TTabSheet* pts[MAXTABS];
const char * ppcTabTitles[] = { "ShortString", "Orders", "Items", "Parts" };
int iTabTitles = sizeof(ppcTabTitles)/sizeof(ppcTabTitles[0]);
const TColor colorPalette[12] = {
  clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia,
  clTeal, clNavy, clMaroon, clLime, clOlive, clPurple};

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    ppc = new TPageControl(this); // The owner (this) will clean this up.
    ppc->Parent = this;
    ppc->Align = alClient;
    for (int i=0;i<iTabTitles;i++)
  {
        pts[i] = new TTabSheet(this); // The owner (this) will clean this up.
        pts[i]->PageControl = ppc;
        pts[i]->Name = AnsiString("pts") + ppcTabTitles[i];
        pts[i]->Caption = ppcTabTitles[i];
        pts[i]->Brush->Color = colorPalette[i];
  }
}
```

Delphi Examples:

```
{
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  Split: TSplitter;
  Files: TFileListBox;
  Dirs: TDirectoryListBox;
begin
  { first add a directory list box to the form }
  Dirs := TDirectoryListBox.Create(Form1);
  { line it up on the left of the form }
  Dirs.Parent := Form1;
  Dirs.Align := alLeft;
  Dirs.Width := Form1.ClientWidth div 3;
{ now add the splitter to divide the directory pane from the file pane }
  Split := TSplitter.Create(Form1);
  Split.Parent := Form1;
  { make sure the splitter is to the right of the directory list box! }
  Split.Left := Dirs.Left + Dirs.Width + 1;
  Split.Align := Dirs.Align; { give it the same alignment as the directory }
  { each pane must be at least one quarter of the form's width }
  Split.MinSize := Form1.ClientWidth div 4;
  { Finally, create the last pane - a file list box }
  Files := TFileListBox.Create(Form1);
  Files.Parent := Form1;
  Files.Align := alClient;
  Dirs.FileList := Files; { link the file list box to the directory list box }
end;
{
Dynamically create a TProgressBar control and align it to
```

```
  the bottom of the form.
  }

  var
    I: Integer;

  procedure TForm1.Button1Click(Sender: TObject);
  var
    ProgressBar: TProgressBar;
  begin
    ProgressBar := TProgressBar.Create(Self);
    with ProgressBar do
    begin
      Parent := Self;
      Align := alBottom;
    end;
  end;
  {
  This example dynamically creates a Page Control, then a
  series of Tab Sheets on the Page Control.
  }
  var
    PageControl1: TPageControl;
    pts: array[0..3] of TTabSheet;
  const
    TabTitles: array[0..3] of ShortString = ('Customer', 'Orders', 'Items', 'Parts' );

  procedure TForm1.FormCreate(Sender: TObject);
  const
    colorpalette: Array[0..11] of TColor = (
      clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia, clTeal,
      clNavy, clMaroon, clLime, clOlive, clPurple);
  var
    i: Integer;
  begin
    PageControl1 := TPageControl.Create(Self);
    PageControl1.Parent := Self;
    PageControl1.Align := alClient;
    for i := Low(TabTitles) to High(TabTitles) do
    begin
      pts[i]:= TTabSheet.Create(PageControl1);
      with pts[i] do
      begin
        PageControl := PageControl1;
        Name := 'ts' + TabTitles[i];
        Caption := TabTitles[i];
        Brush.Color := colorPalette[i];
     end;
    end;
  end;

  procedure TForm1.FormDestroy(Sender: TObject);
  var
    i: Integer;
  begin
    for i := Low(TabTitles) to High(TabTitles) do
    begin
      pts[i].Free;
    end;
    PageControl1.Free;
  end;
```

See Also

Parent

Anchors

64.1.4.5.3 TRibbonApplicationMenuBar.Height Property

Specifies the vertical size of the control in pixels.

Pascal

```
property Height;
```

C++

```
__property Height;
```

Description

Use the Height property to read or change the height of the control.

C++ Examples:

```
/*
This example needs only a blank form.  Three Rich Edit
controls are placed vertically on the form, each contain a
line of text, and the second form has HideSelection set to
false.  Try tabbing between the three Rich Edit controls and
observe the effect on the selected text.
*/
#include <ComCtrls.hpp>

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int I = 0; I <= 2; I++)
  {
    TRichEdit *pRich = new TRichEdit(this);
    pRich->Parent = this;
    pRich->Top = pRich->Height * I;
    pRich->Text = "This text will take the focus.";
    pRich->SelStart = 0;
    pRich->SelLength = pRich->Text.Length();
    if (I == 1)
      pRich->HideSelection = false;
  }
}
```

Delphi Examples:

```
{
This example needs only a blank form. Three Rich Edit
controls are placed vertically on the form, each contain a
line of text, and the second form has HideSelection set to
false.  Try tabbing between the three Rich Edit controls and
observe the effect on the selected text.
}
uses ComCtrls;

procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
  for I := 0 to 2 do
    with TRichEdit.Create(Self) do
    begin
      Parent := Self;
      Top := Height * I;
      Text := 'This text will take the focus.';
      SelStart := 0;
      SelLength := Length(Text);
      if (I = 1) then
        HideSelection := False;
```

```
    end;
end;
```

See Also

ClientHeight

Width

Using TDBListBox and TDBComboBox

64.1.4.5.4 TRibbonApplicationMenuBar.HorzMargin Property

Specifies the width of the horizontal margins in pixels.

Pascal

```
property HorzMargin;
```

C++

```
__property HorzMargin;
```

Description

HorzMargin specifies the width of the action band's left and right horizontal margins in pixels.

See Also

VertMargin

VertSeparator

64.1.4.5.5 TRibbonApplicationMenuBar.Left Property

Specifies the horizontal coordinate of the left edge of a component relative to its parent.

Pascal

```
property Left;
```

C++

```
__property Left;
```

Description

Use the Left property to determine where the left side of the control begins or to reposition the left side of the control.

If the control is contained in another control, the Left and Top properties are relative to the parent control. If the control is contained directly by the form, the property values are relative to the form. For forms, the value of the Left property is relative to the screen in pixels.

C++ Examples:

```
/*
Place a TTimer object in the form and enter Timer1Timer in
the OnTimer event.  Place other controls in the form and
change the active control at runtime.  The following event
handler responds to timer events by moving the active control
one pixel to the right every 100 miliseconds:
*/
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
  Timer1->Interval = 100;
  if (ActiveControl)
    ActiveControl->Left = ActiveControl->Left + 1;
```

```
}
```

Delphi Examples:

```
{
Place a TTimer object in the form and enter Timer1Timer in
the OnTimer event.  Place other controls in the form and
change the active control at runtime.  The following event
handler responds to timer events by moving the active control
one pixel to the right every 100 miliseconds:
}
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Timer1.Interval := 100;
  if ActiveControl <> nil then
    ActiveControl.Left := ActiveControl.Left + 1;
end;
```

See Also

Top

64.1.4.5.6 TRibbonApplicationMenuBar.OptionItems Property

Commands displayed at bottom of Application Menu.

Pascal

```
property OptionItems: TOptionItems;
```

C++

```
__property TOptionItems OptionItems;
```

Description

OptionItems are displayed at the bottom of the Application Menu. Usually this section of commands has Application Exit and Application Options commands.

64.1.4.5.7 TRibbonApplicationMenuBar.RecentItems Property

Recent items displayed in right pane of Application.

Pascal

```
property RecentItems: TOptionItems;
```

C++

```
__property TOptionItems RecentItems;
```

Description

RecentItems are displayed in the right pane of the Application Menu. RecentItems can be added dynamically using one of the RecentItems helper methods. To handle the user clicking on a Recent Item, use the OnRecentItemClick event.

See Also

OnRecentItemClick (see page 3498)

64.1.4.5.8 TRibbonApplicationMenuBar.Spacing Property

Determines where the image and text appear on an action band's rendered control that represents the action client.

Pascal

```
property Spacing;
```

C++

```
__property Spacing;
```

Description

Spacing is used internally to set the number of pixels that should appear between the image specified in an action client's ImageIndex and the text specified in its Caption property.

See Also

Caption

ImageIndex

64.1.4.5.9 TRibbonApplicationMenuBar.Top Property

Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.

Pascal

```
property Top;
```

C++

```
__property Top;
```

Description

Use Top to locate the top of the control or reposition the control to a different Y coordinate. The Top property, like the Left property, is the position of the control relative to its container. Thus, if a control is contained in a TPanel, the Left and Top properties are relative to the panel. If the control is contained directly by the form, it is relative to the form. For forms, the value of the Top property is relative to the screen in pixels.

C++ Examples:

```
/*
This example uses an updown control and a group box on a
form, with several controls contained within the group box.
When the updown control or is clicked, the controls within
the group box are moved in the appropriate direction.
*/
void __fastcall TForm1::UpDown1Click(TObject *Sender, TUDBtnType Button)
{
  int I;
  TControl *ChildControl;

  for (I = 0; I < GroupBox1->ControlCount; I++)
  {
    ChildControl = GroupBox1->Controls[I];
    if (Button == btNext)
      ChildControl->Top = ChildControl->Top + 15;
    else
      ChildControl->Top = ChildControl->Top - 15;
  }
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  const TColor colorarray[5] = {clYellow, clGreen, clBlue, clLime, clFuchsia};
  GroupBox1->Brush->Color = clRed;
  for (int i = 0; i < GroupBox1->ControlCount; i++)
    dynamic_cast<TWinControl *>(GroupBox1->Controls[i])->Brush->Color = colorarray[i];
```

```
}
```

Delphi Examples:

```
{
This example uses an updown control and a group box on a
form, with several controls contained within the group box.
When the updown control is clicked, the controls within
the group box are moved in the appropriate direction.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
const
  colorarray : Array[0..4] of TColor = (clYellow, clGreen, clBlue, clLime, clFuchsia);
begin
   GroupBox1.Brush.Color := clRed;
  for I:= 0 to GroupBox1.ControlCount -1 do
  begin
    TWinControl(GroupBox1.Controls[I]).Brush.Color := colorarray[I];
  end;
end;

procedure TForm1.UpDown1Click(Sender: TObject; Button: TUDBtnType);
var
  I: Integer;
  ChildControl: TControl;
begin
  for I:= 0 to GroupBox1.ControlCount -1 do
  begin
    ChildControl := GroupBox1.Controls[I];
    if Button = btNext then
      ChildControl.Top := ChildControl.Top + 15
    else
      ChildControl.Top := ChildControl.Top - 15;
  end;
end;
```

See Also

Left

64.1.4.5.10 TRibbonApplicationMenuBar.VertMargin Property

Specifies the width of the vertical margins in pixels.

Pascal

```
property VertMargin;
```

C++

```
__property VertMargin;
```

Description

VertMargin specifies the width of the action band's top and bottom vertical margins in pixels.

See Also

HorzMargin

HorzSeparator

64.1.4.5.11 TRibbonApplicationMenuBar.VisibleMenu Property

Pascal

```
property VisibleMenu: TCustomActionMenuBar;
```

C++

```
__property TCustomActionMenuBar VisibleMenu;
```

Description

This is VisibleMenu, a member of class TRibbonApplicationMenuBar.

64.1.4.5.12 TRibbonApplicationMenuBar.VisibleMenuCount Property

Pascal

```
property VisibleMenuCount: Integer;
```

C++

```
__property int VisibleMenuCount;
```

Description

This is VisibleMenuCount, a member of class TRibbonApplicationMenuBar.

64.1.4.5.13 TRibbonApplicationMenuBar.Width Property

Specifies the horizontal size of the control or form in pixels.

Pascal

```
property Width;
```

C++

```
__property Width;
```

Description

Use the Width property to read or change the width of the control.

Note: For tab sheet controls, changing this property at runtime has no effect.

C++ Examples:

```
/*
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
*/

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  TSplitter* ps = new TSplitter(Form1); // These objects will all be cleaned up by the owner
(Form1).
  TFileListBox* pflb = new TFileListBox(Form1); // The parent (Form1) will clean up these
```

```
  objects.
    TDirectoryListBox* pdlb = new TDirectoryListBox(Form1);

    // Line up the directory list box on the left of the form.
    pdlb->Parent = Form1;
    pdlb->Align = alLeft;
    pdlb->Width = Form1->ClientWidth/3;

    // Now use the splitter to divide the directory pane from the file pane.
    ps->Parent = Form1;
    ps->Left = pdlb->Left + pdlb->Width + 1; // Move to right of directory list.
    ps->Align = pdlb->Align; // Give same alignment as directory list.

    // Each pane must be at least one quarter of the form's width.
    ps->MinSize = Form1->ClientWidth/4;

    // Finally, create the last pane - a file list box.
    pflb->Parent = Form1;
    pflb->Align = alClient;
    pdlb->FileList = pflb; // Link the file list box to the directory list box.
}
```

Delphi Examples:

```
{
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  Split: TSplitter;
  Files: TFileListBox;
  Dirs: TDirectoryListBox;
begin
  { first add a directory list box to the form }
  Dirs := TDirectoryListBox.Create(Form1);
  { line it up on the left of the form }
  Dirs.Parent := Form1;
  Dirs.Align := alLeft;
  Dirs.Width := Form1.ClientWidth div 3;
{ now add the splitter to divide the directory pane from the file pane }
  Split := TSplitter.Create(Form1);
  Split.Parent := Form1;
  { make sure the splitter is to the right of the directory list box! }
  Split.Left := Dirs.Left + Dirs.Width + 1;
  Split.Align := Dirs.Align; { give it the same alignment as the directory }
  { each pane must be at least one quarter of the form's width }
  Split.MinSize := Form1.ClientWidth div 4;
  { Finally, create the last pane – a file list box }
  Files := TFileListBox.Create(Form1);
  Files.Parent := Form1;
  Files.Align := alClient;
  Dirs.FileList := Files; { link the file list box to the directory list box }
end;
```

See Also

ClientWidth

Height

Left

64.1.5 TRibbonApplicationPopupMenu Class

Ddisplay popup menu when Application Button clicked.

Class Hierarchy

Pascal

```
TRibbonApplicationPopupMenu = class(TCustomActionPopupMenu);
```

C++

```
class TRibbonApplicationPopupMenu : public TCustomActionPopupMenu;
```

File

RibbonActnMenus

Description

TRibbonApplicationPopupMenu is used to display the popup menu when the Application Button has been clicked. The Application Popup Menu is a two paned menu that shows a normal list of commands on the left pane, as well as a specialized list of commands on the right pane. Additional commands can also be displayed in the bottom border of the menu.

64.1.5.1 TRibbonApplicationPopupMenu Members

The following tables list the members exposed by TRibbonApplicationPopupMenu.

Protected Methods

	Name	Description
	AlignControls (see page 3963)	Aligns any controls for which the control is the parent within a specified area of the control.
	CMTextchanged (see page 3963)	This is CMTextchanged, a member of class TRibbonApplicationPopupMenu.
	CreateControls (see page 3963)	This is CreateControls, a member of class TRibbonApplicationPopupMenu.
	CreateOptionsItems (see page 3964)	This is CreateOptionsItems, a member of class TRibbonApplicationPopupMenu.
	CreateRecentDocuments (see page 3964)	This is CreateRecentDocuments, a member of class TRibbonApplicationPopupMenu.
	CreateWnd (see page 3964)	Creates the underlying window.
	CustomAlignPosition (see page 3965)	Returns aligned position of custom-aligned child control.
	DoDropActions (see page 3965)	This is DoDropActions, a member of class TRibbonApplicationPopupMenu.
	DoDropItem (see page 3966)	This is DoDropItem, a member of class TRibbonApplicationPopupMenu.
	DrawBackground (see page 3966)	This is DrawBackground, a member of class TRibbonApplicationPopupMenu.
	GetBorderHeight (see page 3966)	This is GetBorderHeight, a member of class TRibbonApplicationPopupMenu.
	NCPaint (see page 3967)	This is NCPaint, a member of class TRibbonApplicationPopupMenu.
	PositionPopup (see page 3967)	This is PositionPopup, a member of class TRibbonApplicationPopupMenu.
	RibbonStyle (see page 3967)	This is RibbonStyle, a member of class TRibbonApplicationPopupMenu.
	SetColorMap (see page 3967)	This is SetColorMap, a member of class TRibbonApplicationPopupMenu.

Public Methods

	Name	Description
	Create (see page 3962)	Creates and initializes a TCustomActionPopupMenu instance.
	Destroy (see page 3962)	Destroys an instance of TCustomActionMenuBar.

	DoDropCategory (see page 3966)	This is DoDropCategory, a member of class TRibbonApplicationPopupMenu.
	GetRibbon (see page 3966)	This is GetRibbon, a member of class TRibbonApplicationPopupMenu.
	SetRibbon (see page 3968)	This is SetRibbon, a member of class TRibbonApplicationPopupMenu.

64.1.5.2 TRibbonApplicationPopupMenu.Create Constructor

Creates and initializes a TCustomActionPopupMenu instance.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonApplicationPopupMenu(TComponent * AOwner);
```

Description

Create generates a TCustomActionPopupMenu and initializes its properties.

```
__fastcall virtual TCustomActionMenuBar (Classes::TComponentTComponent_Object* AOwner);
__fastcall TCustomActionMenuBar (HWND ParentWindow) :
Controls::TWinControlTWinControl_Create(ParentWindow) { }
```

64.1.5.3 TRibbonApplicationPopupMenu.Destroy Destructor

Destroys an instance of TCustomActionMenuBar.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRibbonApplicationPopupMenu();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TCustomActionMenuBar reference is not nil, and only then calls Destroy.

See Also

System::TObject::Free

Destroy

64.1.5.4 TRibbonApplicationPopupMenu Methods

The methods of the TRibbonApplicationPopupMenu class are listed here.

Protected Methods

	Name	Description
	AlignControls (see page 3963)	Aligns any controls for which the control is the parent within a specified area of the control.
	CMTextchanged (see page 3963)	This is CMTextchanged, a member of class TRibbonApplicationPopupMenu.
	CreateControls (see page 3963)	This is CreateControls, a member of class TRibbonApplicationPopupMenu.
	CreateOptionsItems (see page 3964)	This is CreateOptionsItems, a member of class TRibbonApplicationPopupMenu.
	CreateRecentDocuments (see page 3964)	This is CreateRecentDocuments, a member of class TRibbonApplicationPopupMenu.
	CreateWnd (see page 3964)	Creates the underlying window.

	CustomAlignPosition (see page 3965)	Returns aligned position of custom-aligned child control.
	DoDropActions (see page 3965)	This is DoDropActions, a member of class TRibbonApplicationPopupMenu.
	DoDropItem (see page 3966)	This is DoDropItem, a member of class TRibbonApplicationPopupMenu.
	DrawBackground (see page 3966)	This is DrawBackground, a member of class TRibbonApplicationPopupMenu.
	GetBorderHeight (see page 3966)	This is GetBorderHeight, a member of class TRibbonApplicationPopupMenu.
	NCPaint (see page 3967)	This is NCPaint, a member of class TRibbonApplicationPopupMenu.
	PositionPopup (see page 3967)	This is PositionPopup, a member of class TRibbonApplicationPopupMenu.
	RibbonStyle (see page 3967)	This is RibbonStyle, a member of class TRibbonApplicationPopupMenu.
	SetColorMap (see page 3967)	This is SetColorMap, a member of class TRibbonApplicationPopupMenu.

Public Methods

	Name	Description
	DoDropCategory (see page 3966)	This is DoDropCategory, a member of class TRibbonApplicationPopupMenu.
	GetRibbon (see page 3966)	This is GetRibbon, a member of class TRibbonApplicationPopupMenu.
	SetRibbon (see page 3968)	This is SetRibbon, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.1 TRibbonApplicationPopupMenu.AlignControls Method

Aligns any controls for which the control is the parent within a specified area of the control.

Pascal

```
procedure AlignControls(AControl: TControl; var Rect: TRect); override;
```

C++

```
virtual __fastcall AlignControls(TControl * AControl, TRect Rect);
```

Description

Use AlignControls to align all controls within an area of a form, panel, group box, scroll box, or any container control. Specify the area in which to align the controls as the value of the Rect parameter.

AlignControls uses the Align property value for each child control to determine how to align it. The AControl parameter can be nil (Delphi) or NULL (C++). If you specify a control in AControl, that control takes precedence in alignment over other, similarly-aligned controls.

See Also

Align

64.1.5.4.2 TRibbonApplicationPopupMenu.CMTextchanged Method

Pascal

```
procedure CMTextchanged(var Message: TMessage); message CM_TEXTCHANGED;
```

C++

```
__fastcall CMTextchanged(TMessage Message);
```

Description

This is CMTextchanged, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.3 TRibbonApplicationPopupMenu.CreateControls Method

Pascal

```
procedure CreateControls; override;
```

C++

```
virtual __fastcall CreateControls();
```

Description

This is CreateControls, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.4 TRibbonApplicationPopupMenu.CreateOptionsItems Method

Pascal

```
procedure CreateOptionsItems; virtual;
```

C++

```
virtual __fastcall CreateOptionsItems();
```

Description

This is CreateOptionsItems, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.5 TRibbonApplicationPopupMenu.CreateRecentDocuments Method

Pascal

```
procedure CreateRecentDocuments; virtual;
```

C++

```
virtual __fastcall CreateRecentDocuments();
```

Description

This is CreateRecentDocuments, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.6 TRibbonApplicationPopupMenu.CreateWnd Method

Creates the underlying window.

Pascal

```
procedure CreateWnd; override;
```

C++

```
virtual __fastcall CreateWnd();
```

Description

CreateWnd is called automatically when the control is first created or when the underlying screen object must be destroyed and recreated to reflect property changes. Override CreateWnd to pass additional initialization messages to the screen object when it is created.

See Also

CreateParams

CreateWindowHandle

Perform

DestroyWnd

64.1.5.4.7 TRibbonApplicationPopupMenu.CustomAlignPosition Method

Returns aligned position of custom-aligned child control.

Pascal

```
procedure CustomAlignPosition(Control: TControl; var NewLeft: Integer; var NewTop: Integer;
var NewWidth: Integer; var NewHeight: Integer; var AlignRect: TRect; AlignInfo: TAlignInfo);
override;
```

C++

```
virtual __fastcall CustomAlignPosition(TControl * Control, int NewLeft, int NewTop, int
NewWidth, int NewHeight, TRect AlignRect, TAlignInfo AlignInfo);
```

Description

CustomAlignPosition determines the specific position of custom-aligned child controls. CustomAlignPosition is called once for each child control with an Align property of **alCustom**, in an order previously determined by calls to CustomAlignInsertBefore.

CustomAlignPosition triggers the OnAlignPosition event. If this event is defined, CustomAlignPosition uses the alignment parameters it obtains from OnAlignPosition. Defining this event allows users to set the alignment parameters without overriding CustomAlignPosition.

These are the parameters:

Control	The child control being placed.
NewLeft | NewTop | NewWidth | NewHeight	Location and size of child control, as determined by container's size and the Anchors TControl_Anchors constraints for the control. CustomAlignPosition can modify these values to reposition the control.
AlignRect	The client area in which the control is aligned. CustomAlignPosition can modify this value.
AlignInfo	Alignment information in a TAlignInfo.
As implemented in TWinControl, CustomAlignPosition does nothing.	

See Also

CustomAlignInsertBefore

Align

OnAlignPosition

TAlignInfo

64.1.5.4.8 TRibbonApplicationPopupMenu.DoDropActions Method

Pascal

```
procedure DoDropActions(Source: TActionDragObject; const X: Integer; const Y: Integer);
override;
```

C++

```
virtual __fastcall DoDropActions(TActionDragObject Source, const int X, const int Y);
```

Description

This is DoDropActions, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.9 TRibbonApplicationPopupMenu.DoDropCategory Method

Pascal

```
procedure DoDropCategory(Source: TCategoryDragObject; const X: Integer; const Y: Integer);
override;
```

C++

```
virtual __fastcall DoDropCategory(TCategoryDragObject Source, const int X, const int Y);
```

Description

This is DoDropCategory, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.10 TRibbonApplicationPopupMenu.DoDropItem Method

Pascal

```
procedure DoDropItem(Source: TActionItemDragObject; const X: Integer; const Y: Integer);
override;
```

C++

```
virtual __fastcall DoDropItem(TActionItemDragObject Source, const int X, const int Y);
```

Description

This is DoDropItem, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.11 TRibbonApplicationPopupMenu.DrawBackground Method

Pascal

```
procedure DrawBackground; override;
```

C++

```
virtual __fastcall DrawBackground();
```

Description

This is DrawBackground, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.12 TRibbonApplicationPopupMenu.GetBorderHeight Method

Pascal

```
function GetBorderHeight: Integer; virtual;
```

C++

```
virtual __fastcall int GetBorderHeight();
```

Description

This is GetBorderHeight, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.13 TRibbonApplicationPopupMenu.GetRibbon Method

Pascal

```
function GetRibbon: TWinControl;
```

C++

```
__fastcall TWinControl * GetRibbon();
```

Description

This is GetRibbon, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.14 TRibbonApplicationPopupMenu.NCPaint Method

Pascal

```
procedure NCPaint(DC: HDC); override;
```

C++

```
virtual __fastcall NCPaint(HDC DC);
```

Description

This is NCPaint, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.15 TRibbonApplicationPopupMenu.PositionPopup Method

Pascal

```
procedure PositionPopup(AnOwner: TCustomActionBar; ParentItem: TCustomActionControl); override;
```

C++

```
virtual __fastcall PositionPopup(TCustomActionBar AnOwner, TCustomActionControl ParentItem);
```

Description

This is PositionPopup, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.16 TRibbonApplicationPopupMenu.RibbonStyle Method

Pascal

```
function RibbonStyle: TRibbonStyleActionBars;
```

C++

```
__fastcall TRibbonStyleActionBars RibbonStyle();
```

Description

This is RibbonStyle, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.17 TRibbonApplicationPopupMenu.SetColorMap Method

Pascal

```
procedure SetColorMap(const Value: TCustomActionBarColorMap); override;
```

C++

```
virtual __fastcall SetColorMap(const TCustomActionBarColorMap Value);
```

Description

This is SetColorMap, a member of class TRibbonApplicationPopupMenu.

64.1.5.4.18 TRibbonApplicationPopupMenu.SetRibbon Method

Pascal

```
procedure SetRibbon(const Value: TWinControl);
```

C++

```
__fastcall SetRibbon(const TWinControl * Value);
```

Description

This is SetRibbon, a member of class TRibbonApplicationPopupMenu.

64.1.6 TRibbonQATCustomizeMenu Class

Class Hierarchy

Pascal

```
TRibbonQATCustomizeMenu = class(TCustomizeActionToolBar);
```

C++

```
class TRibbonQATCustomizeMenu : public TCustomizeActionToolBar;
```

File

RibbonActnMenus

Description

This is class RibbonActnMenus.TRibbonQATCustomizeMenu.

64.1.6.1 TRibbonQATCustomizeMenu Members

The following tables list the members exposed by TRibbonQATCustomizeMenu.

Protected Methods

	Name	Description
	DisplayShadow (see page 3969)	This is DisplayShadow, a member of class TRibbonQATCustomizeMenu.
	DoAddCustomizeItems (see page 3969)	This is DoAddCustomizeItems, a member of class TRibbonQATCustomizeMenu.
	GetControlClass (see page 3969)	This is GetControlClass, a member of class TRibbonQATCustomizeMenu.
	HideShadow (see page 3969)	This is HideShadow, a member of class TRibbonQATCustomizeMenu.
	NCPaint (see page 3970)	This is NCPaint, a member of class TRibbonQATCustomizeMenu.
	PositionPopup (see page 3970)	This is PositionPopup, a member of class TRibbonQATCustomizeMenu.
	VisibleChanging (see page 3970)	Provides the interface for a method that responds to changes in the control's visibility.

64.1.6.2 TRibbonQATCustomizeMenu Methods

The methods of the TRibbonQATCustomizeMenu class are listed here.

Protected Methods

	Name	Description
	DisplayShadow (see page 3969)	This is DisplayShadow, a member of class TRibbonQATCustomizeMenu.
	DoAddCustomizeItems (see page 3969)	This is DoAddCustomizeItems, a member of class TRibbonQATCustomizeMenu.
	GetControlClass (see page 3969)	This is GetControlClass, a member of class TRibbonQATCustomizeMenu.
	HideShadow (see page 3969)	This is HideShadow, a member of class TRibbonQATCustomizeMenu.
	NCPaint (see page 3970)	This is NCPaint, a member of class TRibbonQATCustomizeMenu.
	PositionPopup (see page 3970)	This is PositionPopup, a member of class TRibbonQATCustomizeMenu.
	VisibleChanging (see page 3970)	Provides the interface for a method that responds to changes in the control's visibility.

64.1.6.2.1 TRibbonQATCustomizeMenu.DisplayShadow Method

Pascal

```
procedure DisplayShadow; override;
```

C++

```
virtual __fastcall DisplayShadow();
```

Description

This is DisplayShadow, a member of class TRibbonQATCustomizeMenu.

64.1.6.2.2 TRibbonQATCustomizeMenu.DoAddCustomizeItems Method

Pascal

```
procedure DoAddCustomizeItems(const ActionClient: TActionClient); override;
```

C++

```
virtual __fastcall DoAddCustomizeItems(const TActionClient ActionClient);
```

Description

This is DoAddCustomizeItems, a member of class TRibbonQATCustomizeMenu.

64.1.6.2.3 TRibbonQATCustomizeMenu.GetControlClass Method

Pascal

```
function GetControlClass(AnItem: TActionClientItem): TCustomActionControlClass; override;
```

C++

```
virtual __fastcall TCustomActionControlClass GetControlClass(TActionClientItem AnItem);
```

Description

This is GetControlClass, a member of class TRibbonQATCustomizeMenu.

64.1.6.2.4 TRibbonQATCustomizeMenu.HideShadow Method

Pascal

```
procedure HideShadow; override;
```

C++

```
virtual __fastcall HideShadow();
```

Description

This is HideShadow, a member of class TRibbonQATCustomizeMenu.

64.1.6.2.5 TRibbonQATCustomizeMenu.NCPaint Method

Pascal

```
procedure NCPaint(DC: HDC); override;
```

C++

```
virtual __fastcall NCPaint(HDC DC);
```

Description

This is NCPaint, a member of class TRibbonQATCustomizeMenu.

64.1.6.2.6 TRibbonQATCustomizeMenu.PositionPopup Method

Pascal

```
procedure PositionPopup(AnOwner: TCustomActionBar; ParentItem: TCustomActionControl); override;
```

C++

```
virtual __fastcall PositionPopup(TCustomActionBar AnOwner, TCustomActionControl ParentItem);
```

Description

This is PositionPopup, a member of class TRibbonQATCustomizeMenu.

64.1.6.2.7 TRibbonQATCustomizeMenu.VisibleChanging Method

Provides the interface for a method that responds to changes in the control's visibility.

Pascal

```
procedure VisibleChanging; override;
```

C++

```
virtual __fastcall VisibleChanging();
```

Description

VisibleChanging is called before the value of a control's Visible property changes. As implemented in TControl, it does nothing. Override VisibleChanging to respond to changes in the control's visibility. For example, if a control should not change its visibility, override VisibleChanging to raise an exception when an attempt is made to alter the control's visibility.

See Also

Visible

64.1.7 TRibbonRecentDocuments Class

Class Hierarchy

Pascal

```
TRibbonRecentDocuments = class(TGraphicControl);
```

C++

```
class TRibbonRecentDocuments : public TGraphicControl;
```

File

RibbonActnMenus

Description

This is class RibbonActnMenus.TRibbonRecentDocuments.

64.1.7.1 TRibbonRecentDocuments Members

The following tables list the members exposed by TRibbonRecentDocuments.

Protected Methods

	Name	Description
	Paint (see page 3972)	Render control surface.

Public Methods

	Name	Description
	Create (see page 3971)	Creates an instance of TGraphicControl.

Published Properties

	Name	Description
	Caption (see page 3972)	Specifies a text string that identifies the control to the user.

64.1.7.2 TRibbonRecentDocuments.Create Constructor

Creates an instance of TGraphicControl.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonRecentDocuments(TComponent * AOwner);
```

Description

Calling Create constructs and initializes an instance of TGraphicControl. However, you should never attempt to instantiate a TGraphicControl. This class is intended solely as a base class from which other control classes descend and you should only call Create to instantiate one of these descendants.

AOwner is the component, typically a form, that is responsible for freeing the graphic control. It becomes the value of the Owner property. If you do not explicitly provide an Owner to the constructor (that is, if AOwner is nil (Delphi) or NULL (C++)), your application is responsible for explicitly freeing the graphic control.

Note: If a component's constructor allocates resources or memory, also override the destructor to free those resources.

See Also

Owner

Create

Destroy

64.1.7.3 TRibbonRecentDocuments Methods

The methods of the TRibbonRecentDocuments class are listed here.

Protected Methods

	Name	Description
	Paint (see page 3972)	Render control surface.

64.1.7.3.1 TRibbonRecentDocuments.Paint Method

Render control surface.

Pascal

```
procedure Paint; override;
```

C++

```
virtual __fastcall Paint();
```

Description

The Paint method is called automatically when a lightweight control needs update its display area. Code in the Paint method uses the Canvas property to render the control surface.

As implemented in TGraphicControl, Paint does nothing. Specific rendering logic is provided by Paint methods in descendant control classes.

See Also

Canvas

64.1.7.4 TRibbonRecentDocuments Properties

The properties of the TRibbonRecentDocuments class are listed here.

Published Properties

	Name	Description
	Caption (see page 3972)	Specifies a text string that identifies the control to the user.

64.1.7.4.1 TRibbonRecentDocuments.Caption Property

Specifies a text string that identifies the control to the user.

Pascal

```
property Caption;
```

C++

```
__property Caption;
```

Description

Use Caption to specify the text string that labels the control.

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined

character. To display an ampersand character in the caption, use two ampersands (&&).

Note: Controls that display text use either the Caption property or the Text property to specify the text value. Which property is used depends on the type of control. In general, Caption is used for text that appears as a window title or label, while Text is used for text that appears as the content of a control.

Note: The Caption property inheritted from TControl is a protected property. Access to the Caption requires casting the class to an inheritted one where Caption is a published or public property.

C++ Examples:

```
/*
The following code keeps the size of the label control
constant, even though the length of the label's caption
changes. As a result, the caption of the label is probably
too long to display in the label when the user clicks the
button:
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Label1->AutoSize = false;
  Label1->Caption = "This string is too long to be the caption of this label";
}
/*
This example requires a TPageControl populated by several
TabSheets, a TEdit and a TButton. When the button is
clicked, the caption of the active tab changes to that of
the TEdit.  These lines of code could also be written:
PageControl1.ActivePage.Caption := Edit1.Text;
This would make a TTabSheet variable unnecessary since
PageControl1.->ActivePage is already of type TTabSheet.
*/
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TTabSheet *sheet = PageControl1->ActivePage;
  sheet->Caption = Edit1->Text;
}
/*
This example creates a ToolBar and adds ToolButtons to it.
Include ComCtlrs in the uses clause.
*/
#include <comctrls.hpp>
#include <memory>

void AddButtons(TToolBar *toolBar, TStringList *captions)
{
  for (int i = 0; i < captions->Count; i++)
  {
    TToolButton *button = new TToolButton(toolBar);  // The owner (toolBar) will clean this up.
    button->Parent = toolBar;
    button->Caption = captions->Strings[i];
    if (button->Caption == "|")
      button->Style = tbsSeparator;
    else
      button->Style = tbsButton;
  }
}

TToolBar *myToolBar;

void __fastcall TForm1::FormCreate(TObject *Sender)
{
```

```
  myToolBar = new TToolBar(this); // The owner (this) will clean this up.
  myToolBar->Parent = this;
//  TStringList *captions = new TStringList();
  std::auto_ptr<TStringList> captions(new TStringList());

  captions->Add("New");
  captions->Add("Save");
  captions->Add("|");
  captions->Add("Cut");
  captions->Add("Copy");
  captions->Add("Paste");
  AddButtons(myToolBar, captions.get());
//  delete captions;
  myToolBar->ShowCaptions = True;
  myToolBar->Height = 40;
}
```

Delphi Examples:

```
{
The following code keeps the size of the label control
constant, even though the length of the label's caption
changes. As a result, the caption of the label is probably
too long to display in the label when the user clicks the
button:
}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.AutoSize := False;
  Label1.Caption :=
    'This string is too long to be the caption of this label';
end;
{
This example requires a TPageControl populated by several
TabSheets, a TEdit and a TButton. When the button is
clicked, the caption of the active tab changes to that of
the TEdit.  These lines of code could also be written:
PageControl1.ActivePage.Caption := Edit1.Text;
This would obviate the need for a TTabSheet variable since
PageControl1.->ActivePage is already of type TTabSheet.
}
procedure TForm1.Button1Click(Sender: TObject);
var
  sheet: TTabSheet;
begin
  sheet := PageControl1.ActivePage;
  sheet.Caption := Edit1.Text;
end;
{
This example creates a ToolBar and adds ToolButtons to it.
Include ComCtlrs in the uses clause.
}
procedure AddButtons(ToolBar: TToolBar;
  Const ButtonCaptions: array of String);
var
  I: Integer;
begin
  for I := 0 to High(ButtonCaptions) do
    begin
      with TToolButton.Create(ToolBar) do
      begin
        Parent := ToolBar;
        Caption := ButtonCaptions[I];
        if (ButtonCaptions[I] = '|') then
          Style := tbsSeparator
        else
          Style := tbsButton;
```

```
      end;
    end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  ToolBar1 := TToolBar.Create(Self);
  ToolBar1.Parent := Self;
  AddButtons(ToolBar1, ['New', 'Save', '|', 'Cut', 'Copy', 'Paste']);
  ToolBar1.ShowCaptions := True;
  ToolBar1.Height := 40;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  I: Integer;
begin
  for I := ToolBar1.ButtonCount - 1 downto 0 do
    ToolBar1.Buttons[I].Free;
  ToolBar1.Free;
end;
```

64.1.8 TRibbonStyleMainPopupMenu Class

Class Hierarchy

Pascal

```
TRibbonStyleMainPopupMenu = class(TCustomActionPopupMenu);
```

C++

```
class TRibbonStyleMainPopupMenu : public TCustomActionPopupMenu;
```

File

RibbonActnMenus

Description

This is class RibbonActnMenus.TRibbonStyleMainPopupMenu.

64.1.8.1 TRibbonStyleMainPopupMenu Members

The following tables list the members exposed by TRibbonStyleMainPopupMenu.

Protected Methods

	Name	Description
	NCPaint (see page 3976)	This is NCPaint, a member of class TRibbonStyleMainPopupMenu.
	PositionPopup (see page 3976)	This is PositionPopup, a member of class TRibbonStyleMainPopupMenu.

Public Methods

	Name	Description
	Create (see page 3976)	Creates and initializes a TCustomActionPopupMenu instance.

64.1.8.2 TRibbonStyleMainPopupMenu.Create Constructor

Creates and initializes a TCustomActionPopupMenu instance.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonStyleMainPopupMenu(TComponent * AOwner);
```

Description

Create generates a TCustomActionPopupMenu and initializes its properties.

```
__fastcall virtual TCustomActionMenuBar (Classes::TComponentTComponent_Object* AOwner);
__fastcall TCustomActionMenuBar (HWND ParentWindow) :
Controls::TWinControlTWinControl_Create(ParentWindow) { }
```

64.1.8.3 TRibbonStyleMainPopupMenu Methods

The methods of the TRibbonStyleMainPopupMenu class are listed here.

Protected Methods

	Name	Description
	NCPaint (see page 3976)	This is NCPaint, a member of class TRibbonStyleMainPopupMenu.
	PositionPopup (see page 3976)	This is PositionPopup, a member of class TRibbonStyleMainPopupMenu.

64.1.8.3.1 TRibbonStyleMainPopupMenu.NCPaint Method

Pascal

```
procedure NCPaint(DC: HDC); override;
```

C++

```
virtual __fastcall NCPaint(HDC DC);
```

Description

This is NCPaint, a member of class TRibbonStyleMainPopupMenu.

64.1.8.3.2 TRibbonStyleMainPopupMenu.PositionPopup Method

Pascal

```
procedure PositionPopup(AnOwner: TCustomActionBar; ParentItem: TCustomActionControl); override;
```

C++

```
virtual __fastcall PositionPopup(TCustomActionBar AnOwner, TCustomActionControl ParentItem);
```

Description

This is PositionPopup, a member of class TRibbonStyleMainPopupMenu.

64.1.9 TRibbonStylePopupMenu Class

Class Hierarchy

Pascal

```
TRibbonStylePopupMenu = class(TRibbonActionPopupMenu);
```

C++

```
class TRibbonStylePopupMenu : public TRibbonActionPopupMenu;
```

File

RibbonActnMenus

Description

This is class RibbonActnMenus.TRibbonStylePopupMenu.

64.1.9.1 TRibbonStylePopupMenu Members

The following tables list the members exposed by TRibbonStylePopupMenu.

Protected Methods

	Name	Description
	AlignControls (see page 3978)	Aligns any controls for which the control is the parent within a specified area of the control.
	CanAutoSize (see page 3979)	Indicates how the control can be resized.
	NCPaint (see page 3941)	This is NCPaint, a member of class TRibbonActionPopupMenu.
	WMExitsizemove (see page 3980)	This is WMExitsizemove, a member of class TRibbonStylePopupMenu.
	WMNCHitTest (see page 3980)	This is WMNCHitTest, a member of class TRibbonStylePopupMenu.

Public Methods

	Name	Description
	Create (see page 3977)	Creates and initializes a TCustomActionPopupMenu instance.
	CreateControl (see page 3979)	This is CreateControl, a member of class TRibbonStylePopupMenu.
	Destroy (see page 3978)	Destroys an instance of TCustomActionMenuBar.
	TrackMenu (see page 3942)	Executes the action associated with the menu item.

Public Properties

	Name	Description
	GalleryResize (see page 3980)	This is GalleryResize, a member of class TRibbonStylePopupMenu.
	SelectFirstItem (see page 3942)	This is SelectFirstItem, a member of class TRibbonActionPopupMenu.

64.1.9.2 TRibbonStylePopupMenu.Create Constructor

Creates and initializes a TCustomActionPopupMenu instance.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRibbonStylePopupMenu(TComponent * AOwner);
```

Description

Create generates a TCustomActionPopupMenu and initializes its properties.

```
__fastcall virtual TCustomActionMenuBar (Classes::TComponentTComponent_Object* AOwner);
__fastcall TCustomActionMenuBar (HWND ParentWindow) :
Controls::TWinControlTWinControl_Create(ParentWindow) { }
```

64.1.9.3 TRibbonStylePopupMenu.Destroy Destructor

Destroys an instance of TCustomActionMenuBar.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRibbonStylePopupMenu();
```

Description

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the TCustomActionMenuBar reference is not nil, and only then calls Destroy.

See Also

System::TObject::Free

Destroy

64.1.9.4 TRibbonStylePopupMenu Methods

The methods of the TRibbonStylePopupMenu class are listed here.

Protected Methods

	Name	Description
	AlignControls (see page 3978)	Aligns any controls for which the control is the parent within a specified area of the control.
	CanAutoSize (see page 3979)	Indicates how the control can be resized.
	WMExitsizemove (see page 3980)	This is WMExitsizemove, a member of class TRibbonStylePopupMenu.
	WMNCHitTest (see page 3980)	This is WMNCHitTest, a member of class TRibbonStylePopupMenu.

Public Methods

	Name	Description
	CreateControl (see page 3979)	This is CreateControl, a member of class TRibbonStylePopupMenu.

64.1.9.4.1 TRibbonStylePopupMenu.AlignControls Method

Aligns any controls for which the control is the parent within a specified area of the control.

Pascal

```
procedure AlignControls(AControl: TControl; var Rect: TRect); override;
```

C++

```
virtual __fastcall AlignControls(TControl * AControl, TRect Rect);
```

Description

Use AlignControls to align all controls within an area of a form, panel, group box, scroll box, or any container control. Specify the area in which to align the controls as the value of the Rect parameter.

AlignControls uses the Align property value for each child control to determine how to align it. The AControl parameter can be nil (Delphi) or NULL (C++). If you specify a control in AControl, that control takes precedence in alignment over other, similarly-aligned controls.

See Also

Align

64.1.9.4.2 TRibbonStylePopupMenu.CanAutoSize Method

Indicates how the control can be resized.

Pascal

```
function CanAutoSize(var NewWidth: Integer; var NewHeight: Integer): Boolean; override;
```

C++

```
virtual __fastcall Boolean CanAutoSize(int NewWidth, int NewHeight);
```

Description

CanAutoSize is called automatically when the AutoSize property is true and an attempt is made to resize the control. It allows controls to implement the AutoSize property before the resize sequence that begins with the OnCanResize and OnConstrainedResize events and ends with the OnResize event.

When CanAutoSize is called, the NewWidth and NewHeight parameters indicate the proposed new dimensions of the control. CanAutoSize adjusts these values so that the control's size adjusts to its contents. If the adjustment results in values of NewWidth and NewHeight that differ from the control's current width and height, CanAutoSize returns true, indicating that the resize sequence should proceed. If, after adjustment, NewWidth and NewHeight equal Width and Height, CanAutoSize returns false, and the resize sequence is aborted.

As implemented in TWinControl, CanAutoSize returns true, adjusting NewWidth and NewHeight so that all the control's children fit.

See Also

Constraints

ConstrainedResize

AdjustSize

64.1.9.4.3 TRibbonStylePopupMenu.CreateControl Method

Pascal

```
function CreateControl(AnItem: TActionClientItem): TCustomActionControl; override;
```

C++

```
virtual __fastcall TCustomActionControl CreateControl(TActionClientItem AnItem);
```

Description

This is CreateControl, a member of class TRibbonStylePopupMenu.

64.1.9.4.4 TRibbonStylePopupMenu.WMExitsizemove Method

Pascal

```
procedure WMExitsizemove(var Message: TMessage); message WM_EXITSIZEMOVE;
```

C++

```
__fastcall WMExitsizemove(TMessage Message);
```

Description

This is WMExitsizemove, a member of class TRibbonStylePopupMenu.

64.1.9.4.5 TRibbonStylePopupMenu.WMNCHitTest Method

Pascal

```
procedure WMNCHitTest(var Message: TWMNCHitTest); message WM_NCHITTEST;
```

C++

```
__fastcall WMNCHitTest(TWMNCHitTest Message);
```

Description

This is WMNCHitTest, a member of class TRibbonStylePopupMenu.

64.1.9.5 TRibbonStylePopupMenu Properties

The properties of the TRibbonStylePopupMenu class are listed here.

Public Properties

	Name	Description
	GalleryResize (see page 3980)	This is GalleryResize, a member of class TRibbonStylePopupMenu.

64.1.9.5.1 TRibbonStylePopupMenu.GalleryResize Property

Pascal

```
property GalleryResize: TGalleryResize;
```

C++

```
__property TGalleryResize GalleryResize;
```

Description

This is GalleryResize, a member of class TRibbonStylePopupMenu.

65 RibbonGalleryBar Namespace

This is namespace RibbonGalleryBar.

Topics

Name	Description
Classes (see page 3982)	The following table lists classes in this documentation.

Classes

	Name	Description
	TCustomRibbonGalleryBar (see page 3982)	This is class RibbonGalleryBar.TCustomRibbonGalleryBar.
	TCustomRibbonGalleryPopupBar (see page 3988)	This is class RibbonGalleryBar.TCustomRibbonGalleryPopupBar.
	TRibbonGalleryBar (see page 3990)	This is class RibbonGalleryBar.TRibbonGalleryBar.
	TRibbonGalleryPopupBar (see page 3992)	This is class RibbonGalleryBar.TRibbonGalleryPopupBar.
	TRibbonPopupGalleryBar (see page 3992)	This is class RibbonGalleryBar.TRibbonPopupGalleryBar.

65.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TCustomRibbonGalleryBar (see page 3982)	This is class RibbonGalleryBar.TCustomRibbonGalleryBar.
	TCustomRibbonGalleryPopupBar (see page 3988)	This is class RibbonGalleryBar.TCustomRibbonGalleryPopupBar.
	TRibbonGalleryBar (see page 3990)	This is class RibbonGalleryBar.TRibbonGalleryBar.
	TRibbonGalleryPopupBar (see page 3992)	This is class RibbonGalleryBar.TRibbonGalleryPopupBar.
	TRibbonPopupGalleryBar (see page 3992)	This is class RibbonGalleryBar.TRibbonPopupGalleryBar.

65.1.1 TCustomRibbonGalleryBar Class

Class Hierarchy

Pascal

```
TCustomRibbonGalleryBar = class(TActionToolBar);
```

C++

```
class TCustomRibbonGalleryBar : public TActionToolBar;
```

File

RibbonGalleryBar

Description

This is class RibbonGalleryBar.TCustomRibbonGalleryBar.

65.1.1.1 TCustomRibbonGalleryBar Members

The following tables list the members exposed by TCustomRibbonGalleryBar.

Protected Methods

	Name	Description
	AlignControls (see page 3983)	Aligns any controls for which the control is the parent within a specified area of the control.

Public Methods

	Name	Description
	Create (see page 3983)	Creates and initializes a TCustomActionToolBar instance.
	CreateControl (see page 3984)	This is CreateControl, a member of class TCustomRibbonGalleryBar.
	DoDropCategory (see page 3984)	This is DoDropCategory, a member of class TCustomRibbonGalleryBar.

Public Properties

	Name	Description
	ItemsPerRow (see page 3988)	This is ItemsPerRow, a member of class TCustomRibbonGalleryBar.

	ShowItemCaptions (see page 3988)	This is ShowItemCaptions, a member of class TCustomRibbonGalleryBar.

Published Properties

	Name	Description
	Align (see page 3984)	Determines how the control aligns within its container (parent control).

65.1.1.2 TCustomRibbonGalleryBar.Create Constructor

Creates and initializes a TCustomActionToolBar instance.

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TCustomRibbonGalleryBar(TComponent * AOwner);
```

Description

Create generates a TCustomActionToolBar and initializes its properties.

```
__fastcall virtual TCustomActionToolBar (Classes::TComponentTComponent_Object* AOwner);
__fastcall (HWND ParentWindow) : Controls::TWinControlTWinControl_Create(ParentWindow) { }
```

65.1.1.3 TCustomRibbonGalleryBar Methods

The methods of the TCustomRibbonGalleryBar class are listed here.

Protected Methods

	Name	Description
	AlignControls (see page 3983)	Aligns any controls for which the control is the parent within a specified area of the control.

Public Methods

	Name	Description
	CreateControl (see page 3984)	This is CreateControl, a member of class TCustomRibbonGalleryBar.
	DoDropCategory (see page 3984)	This is DoDropCategory, a member of class TCustomRibbonGalleryBar.

65.1.1.3.1 TCustomRibbonGalleryBar.AlignControls Method

Aligns any controls for which the control is the parent within a specified area of the control.

Pascal

```
procedure AlignControls(AControl: TControl; var Rect: TRect); override;
```

C++

```
virtual __fastcall AlignControls(TControl * AControl, TRect Rect);
```

Description

Use AlignControls to align all controls within an area of a form, panel, group box, scroll box, or any container control. Specify the area in which to align the controls as the value of the Rect parameter.

AlignControls uses the Align property value for each child control to determine how to align it. The AControl parameter can be nil (Delphi) or NULL (C++). If you specify a control in AControl, that control takes precedence in alignment over other, similarly-aligned controls.

See Also

Align

65.1.1.3.2 TCustomRibbonGalleryBar.CreateControl Method

Pascal

```
function CreateControl(AnItem: TActionClientItem): TCustomActionControl; override;
```

C++

```
virtual __fastcall TCustomActionControl CreateControl(TActionClientItem AnItem);
```

Description

This is CreateControl, a member of class TCustomRibbonGalleryBar.

65.1.1.3.3 TCustomRibbonGalleryBar.DoDropCategory Method

Pascal

```
procedure DoDropCategory(Source: TCategoryDragObject; const X: Integer; const Y: Integer);
override;
```

C++

```
virtual __fastcall DoDropCategory(TCategoryDragObject Source, const int X, const int Y);
```

Description

This is DoDropCategory, a member of class TCustomRibbonGalleryBar.

65.1.1.4 TCustomRibbonGalleryBar Properties

The properties of the TCustomRibbonGalleryBar class are listed here.

Public Properties

	Name	Description
	ItemsPerRow (see page 3988)	This is ItemsPerRow, a member of class TCustomRibbonGalleryBar.
	ShowItemCaptions (see page 3988)	This is ShowItemCaptions, a member of class TCustomRibbonGalleryBar.

Published Properties

	Name	Description
	Align (see page 3984)	Determines how the control aligns within its container (parent control).

65.1.1.4.1 TCustomRibbonGalleryBar.Align Property

Determines how the control aligns within its container (parent control).

Pascal

```
property Align;
```

C++

```
__property Align;
```

Description

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form,

panel, or component that contains the control changes. When the parent is resized, an aligned control also resizes so that it continues to span the top, bottom, left, or right edge of the parent.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel.

Tip: If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

Note: To cause a control to maintain a specified relationship with an edge of its parent, but not necessarily lie along one edge of the parent, use the Anchors property instead.

C++ Examples:

```
/*
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
*/

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  TSplitter* ps = new TSplitter(Form1); // These objects will all be cleaned up by the owner
(Form1).
  TFileListBox* pflb = new TFileListBox(Form1); // The parent (Form1) will clean up these
objects.
  TDirectoryListBox* pdlb = new TDirectoryListBox(Form1);

  // Line up the directory list box on the left of the form.
  pdlb->Parent = Form1;
  pdlb->Align = alLeft;
  pdlb->Width = Form1->ClientWidth/3;

  // Now use the splitter to divide the directory pane from the file pane.
  ps->Parent = Form1;
  ps->Left = pdlb->Left + pdlb->Width + 1; // Move to right of directory list.
  ps->Align = pdlb->Align; // Give same alignment as directory list.

  // Each pane must be at least one quarter of the form's width.
  ps->MinSize = Form1->ClientWidth/4;

  // Finally, create the last pane - a file list box.
  pflb->Parent = Form1;
  pflb->Align = alClient;
  pdlb->FileList = pflb; // Link the file list box to the directory list box.
}
/*
Dynamically create a TProgressBar control and align it to
the bottom of the form.
*/
```

```
#include <ComCtrls.hpp>
#include <Controls.hpp>

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TProgressBar *ProgressBar = new TProgressBar(this);  // The owner will clean this up.
  ProgressBar->Parent = this;
  ProgressBar->Align = alBottom;
  TButton *Button1 = new TButton(this);  // The owner will clean this up.
  Button1->Parent = this;
}
/*
This example dynamically creates a Page Control, then a
series of Tab Sheets on the Page Control.
*/
#include <Comctrls.hpp>
TPageControl* ppc;
const int MAXTABS = 3;
TTabSheet* pts[MAXTABS];
const char * ppcTabTitles[] = { "ShortString", "Orders", "Items", "Parts" };
int iTabTitles = sizeof(ppcTabTitles)/sizeof(ppcTabTitles[0]);
const TColor colorPalette[12] = {
  clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia,
  clTeal, clNavy, clMaroon, clLime, clOlive, clPurple};

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    ppc = new TPageControl(this); // The owner (this) will clean this up.
    ppc->Parent = this;
    ppc->Align = alClient;
    for (int i=0;i<iTabTitles;i++)
  {
        pts[i] = new TTabSheet(this); // The owner (this) will clean this up.
        pts[i]->PageControl = ppc;
        pts[i]->Name = AnsiString("pts") + ppcTabTitles[i];
        pts[i]->Caption = ppcTabTitles[i];
        pts[i]->Brush->Color = colorPalette[i];
  }
}
```

Delphi Examples:

```
{
This example shows how to use a splitter to divide a form
into resizable panes.  All objects on the form are created
dynamically in the OnCreate event handler of the form. To
run this example, add ExtCtrls and FileCtrl to the uses
clause of the form's implementation section.
Note that when creating a splitter dynamically at runtime,
it is important to set its position to the appropriate side
of the control it will resize.
}
procedure TForm1.FormCreate(Sender: TObject);
var
  Split: TSplitter;
  Files: TFileListBox;
  Dirs: TDirectoryListBox;
begin
  { first add a directory list box to the form }
  Dirs := TDirectoryListBox.Create(Form1);
  { line it up on the left of the form }
  Dirs.Parent := Form1;
  Dirs.Align := alLeft;
  Dirs.Width := Form1.ClientWidth div 3;
{ now add the splitter to divide the directory pane from the file pane }
  Split := TSplitter.Create(Form1);
  Split.Parent := Form1;
```

```
    { make sure the splitter is to the right of the directory list box! }
    Split.Left := Dirs.Left + Dirs.Width + 1;
    Split.Align := Dirs.Align; { give it the same alignment as the directory }
    { each pane must be at least one quarter of the form's width }
    Split.MinSize := Form1.ClientWidth div 4;
    { Finally, create the last pane - a file list box }
    Files := TFileListBox.Create(Form1);
    Files.Parent := Form1;
    Files.Align := alClient;
    Dirs.FileList := Files; { link the file list box to the directory list box }
end;
{
Dynamically create a TProgressBar control and align it to
the bottom of the form.
}

var
  I: Integer;

procedure TForm1.Button1Click(Sender: TObject);
var
  ProgressBar: TProgressBar;
begin
  ProgressBar := TProgressBar.Create(Self);
  with ProgressBar do
  begin
    Parent := Self;
    Align := alBottom;
  end;
end;
{
This example dynamically creates a Page Control, then a
series of Tab Sheets on the Page Control.
}
var
  PageControl1: TPageControl;
  pts: array[0..3] of TTabSheet;
const
  TabTitles: array[0..3] of ShortString = ('Customer', 'Orders', 'Items', 'Parts' );

procedure TForm1.FormCreate(Sender: TObject);
const
  colorpalette: Array[0..11] of TColor = (
    clRed, clGreen, clYellow, clBlue, clWhite, clFuchsia, clTeal,
    clNavy, clMaroon, clLime, clOlive, clPurple);
var
  i: Integer;
begin
  PageControl1 := TPageControl.Create(Self);
  PageControl1.Parent := Self;
  PageControl1.Align := alClient;
  for i := Low(TabTitles) to High(TabTitles) do
  begin
    pts[i]:= TTabSheet.Create(PageControl1);
    with pts[i] do
    begin
      PageControl := PageControl1;
      Name := 'ts' + TabTitles[i];
      Caption := TabTitles[i];
      Brush.Color := colorPalette[i];
   end;
  end;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
```

```
  i: Integer;
begin
  for i := Low(TabTitles) to High(TabTitles) do
  begin
    pts[i].Free;
  end;
  PageControl1.Free;
end;
```

See Also

Parent

Anchors

65.1.1.4.2 TCustomRibbonGalleryBar.ItemsPerRow Property

Pascal

```
property ItemsPerRow: Integer;
```

C++

```
__property int ItemsPerRow;
```

Description

This is ItemsPerRow, a member of class TCustomRibbonGalleryBar.

65.1.1.4.3 TCustomRibbonGalleryBar.ShowItemCaptions Property

Pascal

```
property ShowItemCaptions: Boolean;
```

C++

```
__property Boolean ShowItemCaptions;
```

Description

This is ShowItemCaptions, a member of class TCustomRibbonGalleryBar.

65.1.2 TCustomRibbonGalleryPopupBar Class

Class Hierarchy

Pascal

```
TCustomRibbonGalleryPopupBar = class(TRibbonActionPopupMenu);
```

C++

```
class TCustomRibbonGalleryPopupBar : public TRibbonActionPopupMenu;
```

File

RibbonGalleryBar

Description

This is class RibbonGalleryBar.TCustomRibbonGalleryPopupBar.

65.1.2.1 TCustomRibbonGalleryPopupBar Members

The following tables list the members exposed by TCustomRibbonGalleryPopupBar.

Protected Methods

	Name	Description
	AlignControls (see page 3989)	Aligns any controls for which the control is the parent within a specified area of the control.
	CreateControls (see page 3990)	This is CreateControls, a member of class TCustomRibbonGalleryPopupBar.
	DoGetControlClass (see page 3990)	This is DoGetControlClass, a member of class TCustomRibbonGalleryPopupBar.
	NCPaint (see page 3941)	This is NCPaint, a member of class TRibbonActionPopupMenu.
	SetColorMap (see page 3990)	This is SetColorMap, a member of class TCustomRibbonGalleryPopupBar.

Public Methods

	Name	Description
	CreateControl (see page 3941)	This is CreateControl, a member of class TRibbonActionPopupMenu.
	TrackMenu (see page 3942)	Executes the action associated with the menu item.

Public Properties

	Name	Description
	SelectFirstItem (see page 3942)	This is SelectFirstItem, a member of class TRibbonActionPopupMenu.

65.1.2.2 TCustomRibbonGalleryPopupBar Methods

The methods of the TCustomRibbonGalleryPopupBar class are listed here.

Protected Methods

	Name	Description
	AlignControls (see page 3989)	Aligns any controls for which the control is the parent within a specified area of the control.
	CreateControls (see page 3990)	This is CreateControls, a member of class TCustomRibbonGalleryPopupBar.
	DoGetControlClass (see page 3990)	This is DoGetControlClass, a member of class TCustomRibbonGalleryPopupBar.
	SetColorMap (see page 3990)	This is SetColorMap, a member of class TCustomRibbonGalleryPopupBar.

65.1.2.2.1 TCustomRibbonGalleryPopupBar.AlignControls Method

Aligns any controls for which the control is the parent within a specified area of the control.

Pascal

```
procedure AlignControls(AControl: TControl; var Rect: TRect); override;
```

C++

```
virtual __fastcall AlignControls(TControl * AControl, TRect Rect);
```

Description

Use AlignControls to align all controls within an area of a form, panel, group box, scroll box, or any container control. Specify the area in which to align the controls as the value of the Rect parameter.

AlignControls uses the Align property value for each child control to determine how to align it. The AControl parameter can be nil (Delphi) or NULL (C++). If you specify a control in AControl, that control takes precedence in alignment over other, similarly-aligned controls.

See Also

Align

65.1.2.2.2 TCustomRibbonGalleryPopupBar.CreateControls Method

Pascal

```
procedure CreateControls; override;
```

C++

```
virtual __fastcall CreateControls();
```

Description

This is CreateControls, a member of class TCustomRibbonGalleryPopupBar.

65.1.2.2.3 TCustomRibbonGalleryPopupBar.DoGetControlClass Method

Pascal

```
function DoGetControlClass(AnItem: TActionClientItem): TCustomActionControlClass; override;
```

C++

```
virtual __fastcall TCustomActionControlClass DoGetControlClass(TActionClientItem AnItem);
```

Description

This is DoGetControlClass, a member of class TCustomRibbonGalleryPopupBar.

65.1.2.2.4 TCustomRibbonGalleryPopupBar.SetColorMap Method

Pascal

```
procedure SetColorMap(const Value: TCustomActionBarColorMap); override;
```

C++

```
virtual __fastcall SetColorMap(const TCustomActionBarColorMap Value);
```

Description

This is SetColorMap, a member of class TCustomRibbonGalleryPopupBar.

65.1.3 TRibbonGalleryBar Class

Class Hierarchy

Pascal

```
TRibbonGalleryBar = class(TCustomRibbonGalleryBar);
```

C++

```
class TRibbonGalleryBar : public TCustomRibbonGalleryBar;
```

File

RibbonGalleryBar

Description

This is class RibbonGalleryBar.TRibbonGalleryBar.

65.1.3.1 TRibbonGalleryBar Members

The following tables list the members exposed by TRibbonGalleryBar.

Protected Methods

	Name	Description
	AlignControls (see page 3983)	Aligns any controls for which the control is the parent within a specified area of the control.

Public Methods

	Name	Description
	Create (see page 3983)	Creates and initializes a TCustomActionToolBar instance.
	CreateControl (see page 3984)	This is CreateControl, a member of class TCustomRibbonGalleryBar.
	DoDropCategory (see page 3984)	This is DoDropCategory, a member of class TCustomRibbonGalleryBar.

Published Properties

	Name	Description
	Align (see page 3984)	Determines how the control aligns within its container (parent control).
	Items (see page 3991)	Lists the action clients that correspond to the child controls of this action band.
	ItemsPerRow (see page 3992)	This is ItemsPerRow, a member of class TRibbonGalleryBar.
	ShowItemCaptions (see page 3992)	This is ShowItemCaptions, a member of class TRibbonGalleryBar.

65.1.3.2 TRibbonGalleryBar Properties

The properties of the TRibbonGalleryBar class are listed here.

Published Properties

	Name	Description
	Items (see page 3991)	Lists the action clients that correspond to the child controls of this action band.
	ItemsPerRow (see page 3992)	This is ItemsPerRow, a member of class TRibbonGalleryBar.
	ShowItemCaptions (see page 3992)	This is ShowItemCaptions, a member of class TRibbonGalleryBar.

65.1.3.2.1 TRibbonGalleryBar.Items Property

Lists the action clients that correspond to the child controls of this action band.

Pascal

```
property Items;
```

C++

```
__property Items;
```

Description

Items is used internally to access the components that the action manager lists to represent the child controls of this action band. Each action client object in the Items collection corresponds to one of the controls that the ActionControls property lists. The entire collection of items belongs to the action client specified by this action band's ActionClient property.

See Also

ActionControls

ActionClient

ContextBar

65.1.3.2.2 TRibbonGalleryBar.ItemsPerRow Property

Pascal

```
property ItemsPerRow: Integer;
```

C++

```
__property int ItemsPerRow;
```

Description

This is ItemsPerRow, a member of class TRibbonGalleryBar.

65.1.3.2.3 TRibbonGalleryBar.ShowItemCaptions Property

Pascal

```
property ShowItemCaptions: Boolean;
```

C++

```
__property Boolean ShowItemCaptions;
```

Description

This is ShowItemCaptions, a member of class TRibbonGalleryBar.

65.1.4 TRibbonGalleryPopupBar Class

Class Hierarchy

Pascal

```
TRibbonGalleryPopupBar = class(TCustomRibbonGalleryPopupBar);
```

C++

```
class TRibbonGalleryPopupBar : public TCustomRibbonGalleryPopupBar;
```

File

RibbonGalleryBar

Description

This is class RibbonGalleryBar.TRibbonGalleryPopupBar.

65.1.5 TRibbonPopupGalleryBar Class

Class Hierarchy

Pascal

```
TRibbonPopupGalleryBar = class(TRibbonGalleryBar);
```

C++

```
class TRibbonPopupGalleryBar : public TRibbonGalleryBar;
```

File

RibbonGalleryBar

Description

This is class RibbonGalleryBar.TRibbonPopupGalleryBar.

65.1.5.1 TRibbonPopupGalleryBar Members

The following tables list the members exposed by TRibbonPopupGalleryBar.

Protected Methods

	Name	Description
	AlignControls (see page 3983)	Aligns any controls for which the control is the parent within a specified area of the control.
	CreateParams (see page 3994)	Initializes a window-creation parameter data structure.

Public Methods

	Name	Description
	Assign (see page 3994)	Copies the contents of another, similar object.
	Create (see page 3983)	Creates and initializes a TCustomActionToolBar instance.
	CreateControl (see page 3984)	This is CreateControl, a member of class TCustomRibbonGalleryBar.
	DisplayGallery (see page 3995)	This is DisplayGallery, a member of class TRibbonPopupGalleryBar.
	DoDropCategory (see page 3984)	This is DoDropCategory, a member of class TCustomRibbonGalleryBar.

Public Properties

	Name	Description
	ItemsPerRow (see page 3988)	This is ItemsPerRow, a member of class TCustomRibbonGalleryBar.
	ShowItemCaptions (see page 3988)	This is ShowItemCaptions, a member of class TCustomRibbonGalleryBar.

Published Properties

	Name	Description
	Align (see page 3984)	Determines how the control aligns within its container (parent control).
	Items (see page 3991)	Lists the action clients that correspond to the child controls of this action band.
	ItemsPerRow (see page 3992)	This is ItemsPerRow, a member of class TRibbonGalleryBar.
	ShowItemCaptions (see page 3992)	This is ShowItemCaptions, a member of class TRibbonGalleryBar.

65.1.5.2 TRibbonPopupGalleryBar Methods

The methods of the TRibbonPopupGalleryBar class are listed here.

Protected Methods

	Name	Description
	CreateParams (see page 3994)	Initializes a window-creation parameter data structure.

Public Methods

	Name	Description
	Assign (see page 3994)	Copies the contents of another, similar object.
	DisplayGallery (see page 3995)	This is DisplayGallery, a member of class TRibbonPopupGalleryBar.

65.1.5.2.1 TRibbonPopupGalleryBar.Assign Method

Copies the contents of another, similar object.

Pascal

```
procedure Assign(Source: TPersistent); override;
```

C++

```
virtual __fastcall Assign(TPersistent * Source);
```

Description

Call Assign to copy the properties or other attributes of one object from another. The standard form of a call to Assign is

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

which tells the Destination object to copy the contents of the Source object to itself.

Most objects override Assign to handle the assignment of properties from similar objects. When overriding Assign, call the inherited method if the destination object can't handle the assignment of properties from the class of the Source parameter.

If no overridden Assign method can handle the assignment of properties from Source, the method implemented in TPersistent calls the source object's AssignTo method. This allows the source object to handle the assignment. If the Source object is nil (Delphi) or NULL (C++), Assign raises an EConvertError exception.

In general, the statement

```
Destination := Source; {Delphi}
Destination = Source; // C++
```

is not the same as the statement

```
Destination.Assign(Source); {Delphi}
Destination->Assign(Source); // C++
```

The assignment operator makes Destination reference the same object as Source, whereas the Assign method copies the contents of the object referenced by Source into the object referenced by Destination.

Note: The types of some properties are also objects. If these properties have write methods that use Assign to set the value of the property, then in these cases the assignment operator does the same thing as the Assign method.

See Also

AssignTo

EConvertError

65.1.5.2.2 TRibbonPopupGalleryBar.CreateParams Method

Initializes a window-creation parameter data structure.

Pascal

```
procedure CreateParams(var Params: TCreateParams); override;
```

C++

```
virtual __fastcall CreateParams(TCreateParams Params);
```

Description

The CreateWnd method calls CreateParams to initialize the parameters it passes to CreateWindowHandle. Override CreateParams to customize the way a control creates its Windows representation. When overriding CreateParams, always call the inherited method first to set the default values, then make any desired adjustments.

CreateParams fills in the parameter that is passed as the value of the Params parameter. This data structure holds information needed when telling Windows to create a window handle. The fields of a TCreateParams record become the parameters to a call to the CreateWindowEx API function.

TWinControl implements CreateParams by setting all the fields of Params to generic base values. A number of the standard controls override CreateParams to change one or more of the default values in Params.

See Also

CreateWindowHandle

CreateWnd

AddBiDiModeExStyle

65.1.5.2.3 TRibbonPopupGalleryBar.DisplayGallery Method

Pascal

```
procedure DisplayGallery(AX: Integer; AY: Integer);
```

C++

```
__fastcall DisplayGallery(int AX, int AY);
```

Description

This is DisplayGallery, a member of class TRibbonPopupGalleryBar.

66 RibbonLunaStyleActnCtrls Namespace

This is namespace RibbonLunaStyleActnCtrls.

Topics

Name	Description
Classes (see page 3998)	The following table lists classes in this documentation.
Variables (see page 4002)	The following table lists variables in this documentation.

Classes

	Name	Description
	TRibbonLunaSkin (see page 3998)	This is class RibbonLunaStyleActnCtrls.TRibbonLunaSkin.
	TRibbonLunaStyleActionBars (see page 3999)	This is class RibbonLunaStyleActnCtrls.TRibbonLunaStyleActionBars.

Variables

Name	Description
RibbonLunaStyle (see page 4002)	This is variable RibbonLunaStyleActnCtrls.RibbonLunaStyle.

66.1 Classes

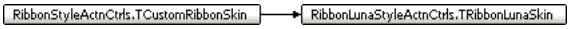
The following table lists classes in this documentation.

Classes

	Name	Description
	TRibbonLunaSkin (see page 3998)	This is class RibbonLunaStyleActnCtrls.TRibbonLunaSkin.
	TRibbonLunaStyleActionBars (see page 3999)	This is class RibbonLunaStyleActnCtrls.TRibbonLunaStyleActionBars.

66.1.1 TRibbonLunaSkin Class

Class Hierarchy



Pascal

```
TRibbonLunaSkin = class(TCustomRibbonSkin);
```

C++

```
class TRibbonLunaSkin : public TCustomRibbonSkin;
```

File

RibbonLunaStyleActnCtrls

Description

This is class RibbonLunaStyleActnCtrls.TRibbonLunaSkin.

66.1.1.1 TRibbonLunaSkin Members

The following tables list the members exposed by TRibbonLunaSkin.

Enumerations

	Name	Description
	TRibbonSkinLoadType (see page 4020)	Indicates how skins are loaded.

Protected Methods

	Name	Description
	DoLoadByFile (see page 4021)	This is DoLoadByFile, a member of class TCustomRibbonSkin.
	DoLoadByResource (see page 4021)	This is DoLoadByResource, a member of class TCustomRibbonSkin.

Public Methods

	Name	Description
	Create (see page 4019)	Create TCustomRibbonSkin (see page 4018) instance.
	Destroy (see page 4019)	Destroy TCustomRibbonSkin (see page 4018) instance.
	GetLoadType (see page 3999)	Set method for loading skins.
	GetLoadTypeValue (see page 3999)	This is GetLoadTypeValue, a member of class TRibbonLunaSkin.
	GetResourceHandle (see page 4021)	Override `GetResourceHandle` to supply skins in separate DLL.

	LoadSkin (see page 4022)	This is LoadSkin, a member of class TCustomRibbonSkin.

Public Properties

	Name	Description
	Bitmap (see page 4022)	This is Bitmap, a member of class TCustomRibbonSkin.

66.1.1.2 TRibbonLunaSkin Methods

The methods of the TRibbonLunaSkin class are listed here.

Public Methods

	Name	Description
	GetLoadType (see page 3999)	Set method for loading skins.
	GetLoadTypeValue (see page 3999)	This is GetLoadTypeValue, a member of class TRibbonLunaSkin.

66.1.1.2.1 TRibbonLunaSkin.GetLoadType Method

Set method for loading skins.

Pascal

```
function GetLoadType: TCustomRibbonSkin; override;
```

C++

```
virtual __fastcall TCustomRibbonSkin GetLoadType();
```

Description

GetLoadType sets the method for loading the skins for use by the application.

66.1.1.2.2 TRibbonLunaSkin.GetLoadTypeValue Method

Pascal

```
function GetLoadTypeValue: string; override;
```

C++

```
virtual __fastcall UnicodeString GetLoadTypeValue();
```

Description

This is GetLoadTypeValue, a member of class TRibbonLunaSkin.

66.1.2 TRibbonLunaStyleActionBars Class

Class Hierarchy

Pascal

```
TRibbonLunaStyleActionBars = class(TRibbonStyleActionBars);
```

C++

```
class TRibbonLunaStyleActionBars : public TRibbonStyleActionBars;
```

File

RibbonLunaStyleActnCtrls

Description

This is class RibbonLunaStyleActnCtrls.TRibbonLunaStyleActionBars.

66.1.2.1 TRibbonLunaStyleActionBars Members

The following tables list the members exposed by TRibbonLunaStyleActionBars.

Public Methods

	Name	Description
	Create (see page 4023)	Create TRibbonStyleActionBars (see page 4022) instance.
	Destroy (see page 4023)	Destroy TRibbonStyleActionBars (see page 4022) instance.
	DrawElement (see page 4024)	This is the overview for the DrawElement method overload.
	GetAddRemoveItemClass (see page 4026)	Returns the base class for ActionBand menus for customizing action bands.
	GetColorMapClass (see page 4000)	Returns the class type of the colormap for an action bar.
	GetControlClass (see page 4027)	Returns the control class type on an action bar item.
	GetPopupClass (see page 4027)	Returns the popup class added to a menu action band.
	GetScrollBtnClass (see page 4027)	This is GetScrollBtnClass, a member of class TRibbonStyleActionBars.
	GetSkinClass (see page 4001)	This is GetSkinClass, a member of class TRibbonLunaStyleActionBars.
	GetStyleName (see page 4001)	Returns the style associated with action manager.

Public Properties

	Name	Description
	RibbonSkin (see page 4028)	Enter a short sentence, starting with a verb, that states the basic purpose of the property.

66.1.2.2 TRibbonLunaStyleActionBars Methods

The methods of the TRibbonLunaStyleActionBars class are listed here.

Public Methods

	Name	Description
	GetColorMapClass (see page 4000)	Returns the class type of the colormap for an action bar.
	GetSkinClass (see page 4001)	This is GetSkinClass, a member of class TRibbonLunaStyleActionBars.
	GetStyleName (see page 4001)	Returns the style associated with action manager.

66.1.2.2.1 TRibbonLunaStyleActionBars.GetColorMapClass Method

Returns the class type of the colormap for an action bar.

Pascal

```
function GetColorMapClass(ActionBar: TCustomActionBar): TCustomColorMapClass; override;
```

C++

```
virtual __fastcall TCustomColorMapClass GetColorMapClass(TCustomActionBar ActionBar);
```

Description

Returns the class type TCustomActionBarColorMap, which is a colormap associated with an action bar. Override this method to customize the appearance of your applications.

66.1.2.2.2 TRibbonLunaStyleActionBars.GetSkinClass Method

Pascal

```
function GetSkinClass: TCustomRibbonSkinClass; override;
```

C++

```
virtual __fastcall TCustomRibbonSkinClass GetSkinClass();
```

Description

This is GetSkinClass, a member of class TRibbonLunaStyleActionBars.

66.1.2.2.3 TRibbonLunaStyleActionBars.GetStyleName Method

Returns the style associated with action manager.

Pascal

```
function GetStyleName: string; override;
```

C++

```
virtual __fastcall UnicodeString GetStyleName();
```

Description

Returns the value of the Style property.

66.2 Variables

The following table lists variables in this documentation.

Variables

Name	Description
RibbonLunaStyle (see page 4002)	This is variable RibbonLunaStyleActnCtrls.RibbonLunaStyle.

66.2.1 RibbonLunaStyleActnCtrls.RibbonLunaStyle Variable

Pascal

```
RibbonLunaStyle: TRibbonLunaStyleActionBars;
```

C++

```
TRibbonLunaStyleActionBars RibbonLunaStyle;
```

File

RibbonLunaStyleActnCtrls

Description

This is variable RibbonLunaStyleActnCtrls.RibbonLunaStyle.

67 RibbonObsidianStyleActnCtrls Namespace

This is namespace RibbonObsidianStyleActnCtrls.

Topics

Name	Description
Classes (see page 4004)	The following table lists classes in this documentation.
Variables (see page 4008)	The following table lists variables in this documentation.

Classes

	Name	Description
	TRibbonObsidianSkin (see page 4004)	This is class RibbonObsidianStyleActnCtrls.TRibbonObsidianSkin.
	TRibbonObsidianStyleActionBars (see page 4005)	This is class RibbonObsidianStyleActnCtrls.TRibbonObsidianStyleActionBars.

Variables

Name	Description
RibbonObsidianStyle (see page 4008)	This is variable RibbonObsidianStyleActnCtrls.RibbonObsidianStyle.

67.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TRibbonObsidianSkin (see page 4004)	This is class RibbonObsidianStyleActnCtrls.TRibbonObsidianSkin.
	TRibbonObsidianStyleActionBars (see page 4005)	This is class RibbonObsidianStyleActnCtrls.TRibbonObsidianStyleActionBars.

67.1.1 TRibbonObsidianSkin Class

Class Hierarchy

Pascal

```
TRibbonObsidianSkin = class(TCustomRibbonSkin);
```

C++

```
class TRibbonObsidianSkin : public TCustomRibbonSkin;
```

File

RibbonObsidianStyleActnCtrls

Description

This is class RibbonObsidianStyleActnCtrls.TRibbonObsidianSkin.

67.1.1.1 TRibbonObsidianSkin Members

The following tables list the members exposed by TRibbonObsidianSkin.

Enumerations

	Name	Description
	TRibbonSkinLoadType (see page 4020)	Indicates how skins are loaded.

Protected Methods

	Name	Description
	DoLoadByFile (see page 4021)	This is DoLoadByFile, a member of class TCustomRibbonSkin.
	DoLoadByResource (see page 4021)	This is DoLoadByResource, a member of class TCustomRibbonSkin.

Public Methods

	Name	Description
	Create (see page 4019)	Create TCustomRibbonSkin (see page 4018) instance.
	Destroy (see page 4019)	Destroy TCustomRibbonSkin (see page 4018) instance.
	GetLoadType (see page 4005)	Set method for loading skins.
	GetLoadTypeValue (see page 4005)	This is GetLoadTypeValue, a member of class TRibbonObsidianSkin.
	GetResourceHandle (see page 4021)	Override `GetResourceHandle` to supply skins in separate DLL.

	LoadSkin (see page 4022)	This is LoadSkin, a member of class TCustomRibbonSkin.

Public Properties

	Name	Description
	Bitmap (see page 4022)	This is Bitmap, a member of class TCustomRibbonSkin.

67.1.1.2 TRibbonObsidianSkin Methods

The methods of the TRibbonObsidianSkin class are listed here.

Public Methods

	Name	Description
	GetLoadType (see page 4005)	Set method for loading skins.
	GetLoadTypeValue (see page 4005)	This is GetLoadTypeValue, a member of class TRibbonObsidianSkin.

67.1.1.2.1 TRibbonObsidianSkin.GetLoadType Method

Set method for loading skins.

Pascal

```
function GetLoadType: TCustomRibbonSkin; override;
```

C++

```
virtual __fastcall TCustomRibbonSkin GetLoadType();
```

Description

GetLoadType sets the method for loading the skins for use by the application.

67.1.1.2.2 TRibbonObsidianSkin.GetLoadTypeValue Method

Pascal

```
function GetLoadTypeValue: string; override;
```

C++

```
virtual __fastcall UnicodeString GetLoadTypeValue();
```

Description

This is GetLoadTypeValue, a member of class TRibbonObsidianSkin.

67.1.2 TRibbonObsidianStyleActionBars Class

Class Hierarchy

Pascal

```
TRibbonObsidianStyleActionBars = class(TRibbonStyleActionBars);
```

C++

```
class TRibbonObsidianStyleActionBars : public TRibbonStyleActionBars;
```

File

RibbonObsidianStyleActnCtrls

Description

This is class RibbonObsidianStyleActnCtrls.TRibbonObsidianStyleActionBars.

67.1.2.1 TRibbonObsidianStyleActionBars Members

The following tables list the members exposed by TRibbonObsidianStyleActionBars.

Public Methods

	Name	Description
	Create (see page 4023)	Create TRibbonStyleActionBars (see page 4022) instance.
	Destroy (see page 4023)	Destroy TRibbonStyleActionBars (see page 4022) instance.
	DrawElement (see page 4024)	This is the overview for the DrawElement method overload.
	GetAddRemoveItemClass (see page 4026)	Returns the base class for ActionBand menus for customizing action bands.
	GetColorMapClass (see page 4006)	Returns the class type of the colormap for an action bar.
	GetControlClass (see page 4027)	Returns the control class type on an action bar item.
	GetPopupClass (see page 4027)	Returns the popup class added to a menu action band.
	GetScrollBtnClass (see page 4027)	This is GetScrollBtnClass, a member of class TRibbonStyleActionBars.
	GetSkinClass (see page 4007)	This is GetSkinClass, a member of class TRibbonObsidianStyleActionBars.
	GetStyleName (see page 4007)	Returns the style associated with action manager.

Public Properties

	Name	Description
	RibbonSkin (see page 4028)	Enter a short sentence, starting with a verb, that states the basic purpose of the property.

67.1.2.2 TRibbonObsidianStyleActionBars Methods

The methods of the TRibbonObsidianStyleActionBars class are listed here.

Public Methods

	Name	Description
	GetColorMapClass (see page 4006)	Returns the class type of the colormap for an action bar.
	GetSkinClass (see page 4007)	This is GetSkinClass, a member of class TRibbonObsidianStyleActionBars.
	GetStyleName (see page 4007)	Returns the style associated with action manager.

67.1.2.2.1 TRibbonObsidianStyleActionBars.GetColorMapClass Method

Returns the class type of the colormap for an action bar.

Pascal

```
function GetColorMapClass(ActionBar: TCustomActionBar): TCustomColorMapClass; override;
```

C++

```
virtual __fastcall TCustomColorMapClass GetColorMapClass(TCustomActionBar ActionBar);
```

Description

Returns the class type TCustomActionBarColorMap, which is a colormap associated with an action bar. Override this method to customize the appearance of your applications.

67.1.2.2.2 TRibbonObsidianStyleActionBars.GetSkinClass Method

Pascal

```
function GetSkinClass: TCustomRibbonSkinClass; override;
```

C++

```
virtual __fastcall TCustomRibbonSkinClass GetSkinClass();
```

Description

This is GetSkinClass, a member of class TRibbonObsidianStyleActionBars.

67.1.2.2.3 TRibbonObsidianStyleActionBars.GetStyleName Method

Returns the style associated with action manager.

Pascal

```
function GetStyleName: string; override;
```

C++

```
virtual __fastcall UnicodeString GetStyleName();
```

Description

Returns the value of the Style property.

67.2 Variables

The following table lists variables in this documentation.

Variables

Name	Description
RibbonObsidianStyle (see page 4008)	This is variable RibbonObsidianStyleActnCtrls.RibbonObsidianStyle.

67.2.1 RibbonObsidianStyleActnCtrls.RibbonObsidianStyle Variable

Pascal

```
RibbonObsidianStyle: TRibbonObsidianStyleActionBars;
```

C++

```
TRibbonObsidianStyleActionBars RibbonObsidianStyle;
```

File

RibbonObsidianStyleActnCtrls

Description

This is variable RibbonObsidianStyleActnCtrls.RibbonObsidianStyle.

68 RibbonSilverStyleActnCtrls Namespace

This is namespace RibbonSilverStyleActnCtrls.

Topics

Name	Description
Classes (see page 4010)	The following table lists classes in this documentation.
Variables (see page 4014)	The following table lists variables in this documentation.

Classes

	Name	Description
	TRibbonSilverSkin (see page 4010)	This is class RibbonSilverStyleActnCtrls.TRibbonSilverSkin.
	TRibbonSilverStyleActionBars (see page 4011)	This is class RibbonSilverStyleActnCtrls.TRibbonSilverStyleActionBars.

Variables

Name	Description
RibbonSilverStyle (see page 4014)	This is variable RibbonSilverStyleActnCtrls.RibbonSilverStyle.

68.1 Classes

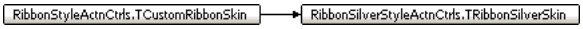
The following table lists classes in this documentation.

Classes

	Name	Description
	TRibbonSilverSkin (see page 4010)	This is class RibbonSilverStyleActnCtrls.TRibbonSilverSkin.
	TRibbonSilverStyleActionBars (see page 4011)	This is class RibbonSilverStyleActnCtrls.TRibbonSilverStyleActionBars.

68.1.1 TRibbonSilverSkin Class

Class Hierarchy



Pascal

```
TRibbonSilverSkin = class(TCustomRibbonSkin);
```

C++

```
class TRibbonSilverSkin : public TCustomRibbonSkin;
```

File

RibbonSilverStyleActnCtrls

Description

This is class RibbonSilverStyleActnCtrls.TRibbonSilverSkin.

68.1.1.1 TRibbonSilverSkin Members

The following tables list the members exposed by TRibbonSilverSkin.

Enumerations

	Name	Description
	TRibbonSkinLoadType (see page 4020)	Indicates how skins are loaded.

Protected Methods

	Name	Description
	DoLoadByFile (see page 4021)	This is DoLoadByFile, a member of class TCustomRibbonSkin.
	DoLoadByResource (see page 4021)	This is DoLoadByResource, a member of class TCustomRibbonSkin.

Public Methods

	Name	Description
	Create (see page 4019)	Create TCustomRibbonSkin (see page 4018) instance.
	Destroy (see page 4019)	Destroy TCustomRibbonSkin (see page 4018) instance.
	GetLoadType (see page 4011)	Set method for loading skins.
	GetLoadTypeValue (see page 4011)	This is GetLoadTypeValue, a member of class TRibbonSilverSkin.
	GetResourceHandle (see page 4021)	Override `GetResourceHandle` to supply skins in separate DLL.

	LoadSkin (see page 4022)	This is LoadSkin, a member of class TCustomRibbonSkin.

Public Properties

	Name	Description
	Bitmap (see page 4022)	This is Bitmap, a member of class TCustomRibbonSkin.

68.1.1.2 TRibbonSilverSkin Methods

The methods of the TRibbonSilverSkin class are listed here.

Public Methods

	Name	Description
	GetLoadType (see page 4011)	Set method for loading skins.
	GetLoadTypeValue (see page 4011)	This is GetLoadTypeValue, a member of class TRibbonSilverSkin.

68.1.1.2.1 TRibbonSilverSkin.GetLoadType Method

Set method for loading skins.

Pascal

```
function GetLoadType: TCustomRibbonSkin; override;
```

C++

```
virtual __fastcall TCustomRibbonSkin GetLoadType();
```

Description

GetLoadType sets the method for loading the skins for use by the application.

68.1.1.2.2 TRibbonSilverSkin.GetLoadTypeValue Method

Pascal

```
function GetLoadTypeValue: string; override;
```

C++

```
virtual __fastcall UnicodeString GetLoadTypeValue();
```

Description

This is GetLoadTypeValue, a member of class TRibbonSilverSkin.

68.1.2 TRibbonSilverStyleActionBars Class

Class Hierarchy

Pascal

```
TRibbonSilverStyleActionBars = class(TRibbonStyleActionBars);
```

C++

```
class TRibbonSilverStyleActionBars : public TRibbonStyleActionBars;
```

File

RibbonSilverStyleActnCtrls

Description

This is class RibbonSilverStyleActnCtrls.TRibbonSilverStyleActionBars.

68.1.2.1 TRibbonSilverStyleActionBars Members

The following tables list the members exposed by TRibbonSilverStyleActionBars.

Public Methods

	Name	Description
	Create (see page 4023)	Create TRibbonStyleActionBars (see page 4022) instance.
	Destroy (see page 4023)	Destroy TRibbonStyleActionBars (see page 4022) instance.
	DrawElement (see page 4024)	This is the overview for the DrawElement method overload.
	GetAddRemoveItemClass (see page 4026)	Returns the base class for ActionBand menus for customizing action bands.
	GetColorMapClass (see page 4012)	Returns the class type of the colormap for an action bar.
	GetControlClass (see page 4027)	Returns the control class type on an action bar item.
	GetPopupClass (see page 4027)	Returns the popup class added to a menu action band.
	GetScrollBtnClass (see page 4027)	This is GetScrollBtnClass, a member of class TRibbonStyleActionBars.
	GetSkinClass (see page 4013)	This is GetSkinClass, a member of class TRibbonSilverStyleActionBars.
	GetStyleName (see page 4013)	Returns the style associated with action manager.

Public Properties

	Name	Description
	RibbonSkin (see page 4028)	Enter a short sentence, starting with a verb, that states the basic purpose of the property.

68.1.2.2 TRibbonSilverStyleActionBars Methods

The methods of the TRibbonSilverStyleActionBars class are listed here.

Public Methods

	Name	Description
	GetColorMapClass (see page 4012)	Returns the class type of the colormap for an action bar.
	GetSkinClass (see page 4013)	This is GetSkinClass, a member of class TRibbonSilverStyleActionBars.
	GetStyleName (see page 4013)	Returns the style associated with action manager.

68.1.2.2.1 TRibbonSilverStyleActionBars.GetColorMapClass Method

Returns the class type of the colormap for an action bar.

Pascal

```
function GetColorMapClass(ActionBar: TCustomActionBar): TCustomColorMapClass; override;
```

C++

```
virtual __fastcall TCustomColorMapClass GetColorMapClass(TCustomActionBar ActionBar);
```

Description

Returns the class type TCustomActionBarColorMap, which is a colormap associated with an action bar. Override this method to customize the appearance of your applications.

68.1.2.2.2 TRibbonSilverStyleActionBars.GetSkinClass Method

Pascal

```
function GetSkinClass: TCustomRibbonSkinClass; override;
```

C++

```
virtual __fastcall TCustomRibbonSkinClass GetSkinClass();
```

Description

This is GetSkinClass, a member of class TRibbonSilverStyleActionBars.

68.1.2.2.3 TRibbonSilverStyleActionBars.GetStyleName Method

Returns the style associated with action manager.

Pascal

```
function GetStyleName: string; override;
```

C++

```
virtual __fastcall UnicodeString GetStyleName();
```

Description

Returns the value of the Style property.

68.2 Variables

The following table lists variables in this documentation.

Variables

Name	Description
RibbonSilverStyle (see page 4014)	This is variable RibbonSilverStyleActnCtrls.RibbonSilverStyle.

68.2.1 RibbonSilverStyleActnCtrls.RibbonSilverStyle Variable

Pascal

```
RibbonSilverStyle: TRibbonSilverStyleActionBars;
```

C++

```
TRibbonSilverStyleActionBars RibbonSilverStyle;
```

File

RibbonSilverStyleActnCtrls

Description

This is variable RibbonSilverStyleActnCtrls.RibbonSilverStyle.

69 RibbonStyleActnCtrls Namespace

This is namespace RibbonStyleActnCtrls.

Topics

Name	Description
Classes (see page 4017)	The following table lists classes in this documentation.
Structs, Records, Enums (see page 4029)	The following table lists structs, records, enums in this documentation.
Types (see page 4053)	The following table lists types in this documentation.

Classes

	Name	Description
	TAlphaBitmap (see page 4017)	Set AlphaFormat property to **afDefined**.
	TCustomRibbonSkin (see page 4018)	Load resources from file or application.
	TRibbonStyleActionBars (see page 4022)	Action bar style.

Structs, Records, Enums

	Name	Description
	TRibbonElement (see page 4029)	Elements of Ribbon (see page 3347) that need to be skinned.
	TSkinApplicationMenu (see page 4030)	Individual skinned items of Application Menu.
	TSkinButton (see page 4032)	Individual skinned items of Button.
	TSkinCheckBox (see page 4036)	This is record RibbonStyleActnCtrls.TSkinCheckBox.
	TSkinComboBox (see page 4037)	Individual skinned items of Combo.
	TSkinForm (see page 4038)	Individual skinned items of Form.
	TSkinGallery (see page 4040)	Individual skinned items of Gallery.
	TSkinMenu (see page 4041)	Individual skinned items of Menu.
	TSkinQuickAccessToolbar (see page 4043)	Individual skinned items of Quick Access Toolbar.
	TSkinRadioButton (see page 4045)	This is record RibbonStyleActnCtrls.TSkinRadioButton.
	TSkinRibbon (see page 4045)	Individual skinned items of Ribbon (see page 3347).
	TSkinRibbonGroup (see page 4046)	Individual skinned items of Group.
	TSkinScrollBar (see page 4048)	Individual skinned items of ScrollBar.
	TSkinSpinEdit (see page 4049)	Individual skinned items of Spinner.
	TSkinStatusBar (see page 4050)	Individual skinned items of Status Bar.
	TSkinTab (see page 4051)	Individual skinned items of Tab.

Types

Name	Description
TCustomRibbonSkinClass (see page 4053)	This is type RibbonStyleActnCtrls.TCustomRibbonSkinClass.

69.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TAlphaBitmap (see page 4017)	Set AlphaFormat property to **afDefined**.
	TCustomRibbonSkin (see page 4018)	Load resources from file or application.
	TRibbonStyleActionBars (see page 4022)	Action bar style.

69.1.1 TAlphaBitmap Class

Set AlphaFormat property to **afDefined**.

Class Hierarchy

Pascal

```
TAlphaBitmap = class(TBitmap);
```

C++

```
class TAlphaBitmap : public TBitmap;
```

File

RibbonStyleActnCtrls

Description

TAlphaBitmap is a TBitmap descendant that automatically sets the AlphaFormat property to **afDefined**.

See Also

TBitmap

AlphaFormat

TAlphaFormat

69.1.1.1 TAlphaBitmap Members

The following tables list the members exposed by TAlphaBitmap.

Public Methods

	Name	Description
	Create (see page 4018)	Create TAlphaBitmap (see page 4017) instance.

69.1.1.2 TAlphaBitmap.Create Constructor

Create TAlphaBitmap (see page 4017) instance.

Pascal

```
constructor Create; override;
```

C++

```
virtual __fastcall TAlphaBitmap();
```

Description

This method creates an TAlphaBitmap (see page 4017) instance.

69.1.2 TCustomRibbonSkin Class

Load resources from file or application.

Class Hierarchy

RibbonStyleActnCtrls.TCustomRibbonSkin

Pascal

```
TCustomRibbonSkin = class;
```

C++

```
class TCustomRibbonSkin;
```

File

RibbonStyleActnCtrls

Description

TCustomRibbonSkin is responsible for loading the resources from a file or application. It also loads the common elements in each skin. Each descendant then modifies the resource or file to be loaded.

See Also

TRibbonSkinLoadType (see page 4020)

69.1.2.1 TCustomRibbonSkin Members

The following tables list the members exposed by TCustomRibbonSkin.

Enumerations

	Name	Description
	TRibbonSkinLoadType (see page 4020)	Indicates how skins are loaded.

Protected Methods

	Name	Description
	DoLoadByFile (see page 4021)	This is DoLoadByFile, a member of class TCustomRibbonSkin.
	DoLoadByResource (see page 4021)	This is DoLoadByResource, a member of class TCustomRibbonSkin.

Public Methods

	Name	Description
	Create (see page 4019)	Create TCustomRibbonSkin (see page 4018) instance.
	Destroy (see page 4019)	Destroy TCustomRibbonSkin (see page 4018) instance.
	GetLoadType (see page 4021)	Set method for loading skins.
	GetLoadTypeValue (see page 4021)	This is GetLoadTypeValue, a member of class TCustomRibbonSkin.
	GetResourceHandle (see page 4021)	Override `GetResourceHandle` to supply skins in separate DLL.
	LoadSkin (see page 4022)	This is LoadSkin, a member of class TCustomRibbonSkin.

Public Properties

	Name	Description
	Bitmap (see page 4022)	This is Bitmap, a member of class TCustomRibbonSkin.

69.1.2.2 TCustomRibbonSkin.Create Constructor

Create TCustomRibbonSkin (see page 4018) instance.

Pascal

```
constructor Create; virtual;
```

C++

```
virtual __fastcall TCustomRibbonSkin();
```

Description

This method creates a TCustomRibbonSkin (see page 4018) instance.

See Also

Destroy (see page 4019)

69.1.2.3 TCustomRibbonSkin.Destroy Destructor

Destroy TCustomRibbonSkin (see page 4018) instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TCustomRibbonSkin();
```

Description

This method destroys a TCustomRibbonSkin (see page 4018) instance.

See Also

Create (see page 4019)

69.1.2.4 TCustomRibbonSkin Enumerations

The enumerations of the TCustomRibbonSkin class are listed here.

Enumerations

	Name	Description
	TRibbonSkinLoadType (see page 4020)	Indicates how skins are loaded.

69.1.2.4.1 RibbonStyleActnCtrls.TCustomRibbonSkin.TRibbonSkinLoadType Enumeration

Indicates how skins are loaded.

Pascal

```
TRibbonSkinLoadType = (
  ltResource,
  ltFile
);
```

C++

```
enum TRibbonSkinLoadType {
  ltResource,
  ltFile
};
```

File

RibbonStyleActnCtrls

Description

TRibbonSkinLoadType enumerates how skins are loaded.

Value	Meaning
ltResource	Skins are loaded as a resource.
ltFile	Skins are loaded from a single file.
ltFolder	Skins are loaded from a specific folder.

See Also

TCustomRibbonSkin (see page 4018)

69.1.2.5 TCustomRibbonSkin Methods

The methods of the TCustomRibbonSkin class are listed here.

Protected Methods

	Name	Description
	DoLoadByFile (see page 4021)	This is DoLoadByFile, a member of class TCustomRibbonSkin.
	DoLoadByResource (see page 4021)	This is DoLoadByResource, a member of class TCustomRibbonSkin.

Public Methods

	Name	Description
	GetLoadType (see page 4021)	Set method for loading skins.
	GetLoadTypeValue (see page 4021)	This is GetLoadTypeValue, a member of class TCustomRibbonSkin.
	GetResourceHandle (see page 4021)	Override `GetResourceHandle` to supply skins in separate DLL.
	LoadSkin (see page 4022)	This is LoadSkin, a member of class TCustomRibbonSkin.

69.1.2.5.1 TCustomRibbonSkin.DoLoadByFile Method

Pascal

```
procedure DoLoadByFile; virtual;
```

C++

```
virtual __fastcall DoLoadByFile();
```

Description

This is DoLoadByFile, a member of class TCustomRibbonSkin.

69.1.2.5.2 TCustomRibbonSkin.DoLoadByResource Method

Pascal

```
procedure DoLoadByResource; virtual;
```

C++

```
virtual __fastcall DoLoadByResource();
```

Description

This is DoLoadByResource, a member of class TCustomRibbonSkin.

69.1.2.5.3 TCustomRibbonSkin.GetLoadType Method

Set method for loading skins.

Pascal

```
function GetLoadType: TRibbonSkinLoadType; virtual;
```

C++

```
virtual __fastcall TRibbonSkinLoadType GetLoadType();
```

Description

GetLoadType sets the method for loading the skins for use by the application.

69.1.2.5.4 TCustomRibbonSkin.GetLoadTypeValue Method

Pascal

```
function GetLoadTypeValue: string; virtual;
```

C++

```
virtual __fastcall UnicodeString GetLoadTypeValue();
```

Description

This is GetLoadTypeValue, a member of class TCustomRibbonSkin.

69.1.2.5.5 TCustomRibbonSkin.GetResourceHandle Method

Override `GetResourceHandle` to supply skins in separate DLL.

Pascal

```
function GetResourceHandle: THandle; virtual;
```

C++

```
virtual __fastcall THandle GetResourceHandle();
```

Description

This method overrides `GetResourceHandle` to supply skins in a separate DLL file.

69.1.2.5.6 TCustomRibbonSkin.LoadSkin Method

Pascal

```
procedure LoadSkin; virtual;
```

C++

```
virtual __fastcall LoadSkin();
```

Description

This is LoadSkin, a member of class TCustomRibbonSkin.

69.1.2.6 TCustomRibbonSkin Properties

The properties of the TCustomRibbonSkin class are listed here.

Public Properties

	Name	Description
	Bitmap (see page 4022)	This is Bitmap, a member of class TCustomRibbonSkin.

69.1.2.6.1 TCustomRibbonSkin.Bitmap Property

Pascal

```
property Bitmap: TAlphaBitmap;
```

C++

```
__property TAlphaBitmap Bitmap;
```

Description

This is Bitmap, a member of class TCustomRibbonSkin.

69.1.3 TRibbonStyleActionBars Class

Action bar style.

Class Hierarchy

Pascal

```
TRibbonStyleActionBars = class(TXPStyleActionBars);
```

C++

```
class TRibbonStyleActionBars : public TXPStyleActionBars;
```

File

RibbonStyleActnCtrls

Description

TRibbonStyleActionBars is an action bar style.

69.1.3.1 TRibbonStyleActionBars Members

The following tables list the members exposed by TRibbonStyleActionBars.

Public Methods

	Name	Description
	Create (see page 4023)	Create TRibbonStyleActionBars (see page 4022) instance.
	Destroy (see page 4023)	Destroy TRibbonStyleActionBars (see page 4022) instance.
	DrawElement (see page 4024)	This is the overview for the DrawElement method overload.
	GetAddRemoveItemClass (see page 4026)	Returns the base class for ActionBand menus for customizing action bands.
	GetColorMapClass (see page 4026)	Returns the class type of the colormap for an action bar.
	GetControlClass (see page 4027)	Returns the control class type on an action bar item.
	GetPopupClass (see page 4027)	Returns the popup class added to a menu action band.
	GetScrollBtnClass (see page 4027)	This is GetScrollBtnClass, a member of class TRibbonStyleActionBars.
	GetSkinClass (see page 4028)	This is GetSkinClass, a member of class TRibbonStyleActionBars.

Public Properties

	Name	Description
	RibbonSkin (see page 4028)	Enter a short sentence, starting with a verb, that states the basic purpose of the property.

69.1.3.2 TRibbonStyleActionBars.Create Constructor

Create TRibbonStyleActionBars (see page 4022) instance.

Pascal

```
constructor Create; virtual;
```

C++

```
virtual __fastcall TRibbonStyleActionBars();
```

Description

This method creates a TRibbonStyleActionBars (see page 4022) instance.

See Also

Destroy (see page 4023)

69.1.3.3 TRibbonStyleActionBars.Destroy Destructor

Destroy TRibbonStyleActionBars (see page 4022) instance.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRibbonStyleActionBars();
```

Description

This method destroys a TRibbonStyleActionBars (see page 4022) instance.

See Also

Create (see page 4023)

69.1.3.4 TRibbonStyleActionBars Methods

The methods of the TRibbonStyleActionBars class are listed here.

Public Methods

	Name	Description
	DrawElement (see page 4024)	This is the overview for the DrawElement method overload.
	GetAddRemoveItemClass (see page 4026)	Returns the base class for ActionBand menus for customizing action bands.
	GetColorMapClass (see page 4026)	Returns the class type of the colormap for an action bar.
	GetControlClass (see page 4027)	Returns the control class type on an action bar item.
	GetPopupClass (see page 4027)	Returns the popup class added to a menu action band.
	GetScrollBtnClass (see page 4027)	This is GetScrollBtnClass, a member of class TRibbonStyleActionBars.
	GetSkinClass (see page 4028)	This is GetSkinClass, a member of class TRibbonStyleActionBars.

69.1.3.4.1 DrawElement Method

This is the overview for the DrawElement method overload.

Overload List

	Name	Description
	TRibbonStyleActionBars.DrawElement (TSkinApplicationMenu, TCanvas, Integer, Integer, Byte) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinApplicationMenu, TCanvas, TRect) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinButton, TCanvas, TRect, Byte) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinCheckBox, TCanvas, TRect) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinComboBox, TCanvas, TRect) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinForm, TCanvas, TRect, Integer) (see page 4025)	Methods to draw ribbon skin element.

	TRibbonStyleActionBars.DrawElement (TSkinGallery, TCanvas, TRect) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinMenu, TCanvas, TRect) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinQuickAccessToolbar, TCanvas, TRect) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinRadioButton, TCanvas, TRect) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinRibbon, TCanvas, TRect, Integer) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinRibbonGroup, TCanvas, TRect) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinScrollBar, TCanvas, TRect) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinSpinEdit, TCanvas, TRect) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinStatusBar, TCanvas, TRect) (see page 4025)	Methods to draw ribbon skin element.
	TRibbonStyleActionBars.DrawElement (TSkinTab, TCanvas, TRect, Byte) (see page 4025)	Methods to draw ribbon skin element.

69.1.3.4.1.1 TRibbonStyleActionBars.DrawElement Method (TSkinApplicationMenu, TCanvas, Integer, Integer, Byte)

Methods to draw ribbon skin element.

Pascal

```
procedure DrawElement(Element: TSkinRibbon; Canvas: TCanvas; Rect: TRect; Height: Integer);
overload;
procedure DrawElement(Element: TSkinForm; Canvas: TCanvas; Rect: TRect; Height: Integer);
overload;
procedure DrawElement(Element: TSkinApplicationMenu; Canvas: TCanvas; Rect: TRect); overload;
procedure DrawElement(Element: TSkinApplicationMenu; Canvas: TCanvas; const X: Integer; const
Y: Integer; const Opacity: Byte); overload;
procedure DrawElement(Element: TSkinTab; Canvas: TCanvas; Rect: TRect; Opacity: Byte = 255);
overload;
procedure DrawElement(Element: TSkinRibbonGroup; Canvas: TCanvas; Rect: TRect); overload;
procedure DrawElement(Element: TSkinMenu; Canvas: TCanvas; Rect: TRect); overload;
procedure DrawElement(Element: TSkinButton; Canvas: TCanvas; Rect: TRect; Opacity: Byte =
255); overload;
procedure DrawElement(Element: TSkinComboBox; Canvas: TCanvas; Rect: TRect); overload;
procedure DrawElement(Element: TSkinSpinEdit; Canvas: TCanvas; Rect: TRect); overload;
procedure DrawElement(Element: TSkinGallery; Canvas: TCanvas; Rect: TRect); overload;
procedure DrawElement(Element: TSkinQuickAccessToolbar; Canvas: TCanvas; Rect: TRect);
overload;
```

```
procedure DrawElement(Element: TSkinScrollBar; Canvas: TCanvas; Rect: TRect); overload;
procedure DrawElement(Element: TSkinStatusBar; Canvas: TCanvas; Rect: TRect); overload;
procedure DrawElement(Element: TSkinCheckBox; Canvas: TCanvas; Rect: TRect); overload;
procedure DrawElement(Element: TSkinRadioButton; Canvas: TCanvas; Rect: TRect); overload;
```

C++

```
__fastcall DrawElement(TSkinRibbon Element, TCanvas Canvas, TRect Rect, int Height);
__fastcall DrawElement(TSkinForm Element, TCanvas Canvas, TRect Rect, int Height);
__fastcall DrawElement(TSkinApplicationMenu Element, TCanvas Canvas, TRect Rect);
__fastcall DrawElement(TSkinApplicationMenu Element, TCanvas Canvas, const int X, const int Y,
const Byte Opacity);
__fastcall DrawElement(TSkinTab Element, TCanvas Canvas, TRect Rect, Byte Opacity = 255);
__fastcall DrawElement(TSkinRibbonGroup Element, TCanvas Canvas, TRect Rect);
__fastcall DrawElement(TSkinMenu Element, TCanvas Canvas, TRect Rect);
__fastcall DrawElement(TSkinButton Element, TCanvas Canvas, TRect Rect, Byte Opacity = 255);
__fastcall DrawElement(TSkinComboBox Element, TCanvas Canvas, TRect Rect);
__fastcall DrawElement(TSkinSpinEdit Element, TCanvas Canvas, TRect Rect);
__fastcall DrawElement(TSkinGallery Element, TCanvas Canvas, TRect Rect);
__fastcall DrawElement(TSkinQuickAccessToolbar Element, TCanvas Canvas, TRect Rect);
__fastcall DrawElement(TSkinScrollBar Element, TCanvas Canvas, TRect Rect);
__fastcall DrawElement(TSkinStatusBar Element, TCanvas Canvas, TRect Rect);
__fastcall DrawElement(TSkinCheckBox Element, TCanvas Canvas, TRect Rect);
__fastcall DrawElement(TSkinRadioButton Element, TCanvas Canvas, TRect Rect);
```

Description

DrawElement are methods to draw a particular ribbon skin element.

The parameters are:

- **Element**: The item to draw from the Element group.
- **Canvas**: Where the element is to be drawn.
- **Rect**: The rectangle for drawing the element.
- **Center**: Set to True to center the element within the given Rectangle.

69.1.3.4.2 TRibbonStyleActionBars.GetAddRemoveItemClass Method

Returns the base class for ActionBand menus for customizing action bands.

Pascal

```
function GetAddRemoveItemClass(ActionBar: TCustomActionBar): TCustomAddRemoveItemClass;
override;
```

C++

```
virtual __fastcall TCustomAddRemoveItemClass GetAddRemoveItemClass(TCustomActionBar ActionBar);
```

Description

Call GetAddRemoveItemClass to customize action bands.

See Also

TCustomAddRemoveItem

69.1.3.4.3 TRibbonStyleActionBars.GetColorMapClass Method

Returns the class type of the colormap for an action bar.

Pascal

```
function GetColorMapClass(ActionBar: TCustomActionBar): TCustomColorMapClass; override;
```

C++

```
virtual __fastcall TCustomColorMapClass GetColorMapClass(TCustomActionBar ActionBar);
```

Description

Returns the class type TCustomActionBarColorMap, which is a colormap associated with an action bar. Override this method to customize the appearance of your applications.

69.1.3.4.4 TRibbonStyleActionBars.GetControlClass Method

Returns the control class type on an action bar item.

Pascal

```
function GetControlClass(ActionBar: TCustomActionBar; AnItem: TActionClientItem):
TCustomActionControlClass; override;
```

C++

```
virtual __fastcall TCustomActionControlClass GetControlClass(TCustomActionBar ActionBar,
TActionClientItem AnItem);
```

Description

Returns the control class type on an item in the action bar, such as a button or menu item.

69.1.3.4.5 TRibbonStyleActionBars.GetPopupClass Method

Returns the popup class added to a menu action band.

Pascal

```
function GetPopupClass(ActionBar: TCustomActionBar): TCustomPopupClass; override;
```

C++

```
virtual __fastcall TCustomPopupClass GetPopupClass(TCustomActionBar ActionBar);
```

Description

Call GetPopupClass to find the popup menu associated with a menu action band. Override GetPopupClass to change which popup class is associated with a menu action band depending on the state of the control.

See Also

TCustomActionPopupMenu

69.1.3.4.6 TRibbonStyleActionBars.GetScrollBtnClass Method

Pascal

```
function GetScrollBtnClass: TCustomToolScrollBtnClass; override;
```

C++

```
virtual __fastcall TCustomToolScrollBtnClass GetScrollBtnClass();
```

Description

This is GetScrollBtnClass, a member of class TRibbonStyleActionBars.

69.1.3.4.7 TRibbonStyleActionBars.GetSkinClass Method

Pascal

```
function GetSkinClass: TCustomRibbonSkinClass; virtual; abstract;
```

C++

```
virtual __fastcall TCustomRibbonSkinClass GetSkinClass() = 0;
```

Description

This is GetSkinClass, a member of class TRibbonStyleActionBars.

69.1.3.5 TRibbonStyleActionBars Properties

The properties of the TRibbonStyleActionBars class are listed here.

Public Properties

	Name	Description
	RibbonSkin (see page 4028)	Enter a short sentence, starting with a verb, that states the basic purpose of the property.

69.1.3.5.1 TRibbonStyleActionBars.RibbonSkin Property

Enter a short sentence, starting with a verb, that states the basic purpose of the property.

Pascal

```
property RibbonSkin: TCustomRibbonSkin;
```

C++

```
__property TCustomRibbonSkin RibbonSkin;
```

Description

The long description elaborates on the short description, containing no more than a paragraph or two.

69.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TRibbonElement (see page 4029)	Elements of Ribbon (see page 3347) that need to be skinned.
	TSkinApplicationMenu (see page 4030)	Individual skinned items of Application Menu.
	TSkinButton (see page 4032)	Individual skinned items of Button.
	TSkinCheckBox (see page 4036)	This is record RibbonStyleActnCtrls.TSkinCheckBox.
	TSkinComboBox (see page 4037)	Individual skinned items of Combo.
	TSkinForm (see page 4038)	Individual skinned items of Form.
	TSkinGallery (see page 4040)	Individual skinned items of Gallery.
	TSkinMenu (see page 4041)	Individual skinned items of Menu.
	TSkinQuickAccessToolbar (see page 4043)	Individual skinned items of Quick Access Toolbar.
	TSkinRadioButton (see page 4045)	This is record RibbonStyleActnCtrls.TSkinRadioButton.
	TSkinRibbon (see page 4045)	Individual skinned items of Ribbon (see page 3347).
	TSkinRibbonGroup (see page 4046)	Individual skinned items of Group.
	TSkinScrollBar (see page 4048)	Individual skinned items of ScrollBar.
	TSkinSpinEdit (see page 4049)	Individual skinned items of Spinner.
	TSkinStatusBar (see page 4050)	Individual skinned items of Status Bar.
	TSkinTab (see page 4051)	Individual skinned items of Tab.

69.2.1 RibbonStyleActnCtrls.TRibbonElement Enumeration

Elements of Ribbon (see page 3347) that need to be skinned.

Pascal

```
TRibbonElement = (
  reRibbon,
  reForm,
  reApplicationMenu,
  reTab,
  reGroup,
  reMenu,
  reButton,
  reCombo,
  reSpinner,
  reGallery,
  reQuickAccessToolbar,
  reScrollBar,
  reStatusBar,
  reCheckBox,
  reRadioButton
);
```

C++

```
enum TRibbonElement {
  reRibbon,
  reForm,
  reApplicationMenu,
  reTab,
```

```
    reGroup,
    reMenu,
    reButton,
    reCombo,
    reSpinner,
    reGallery,
    reQuickAccessToolbar,
    reScrollBar,
    reStatusBar,
    reCheckBox,
    reRadioButton
  };
```

File

RibbonStyleActnCtrls

Description

TRibbonElement enumerates the available elements of the Ribbon (see page 3347) that need to be skinned.

Value	Meaning
reRibbon	Ribbon (see page 3347)
reForm	Form
reApplicationMenu	Application Menu
reTab	Tab
reGroup	Group
reMenu	Menu
reButton	Button
reCombo	Combo
reSpinner	Spinner
reGallery	Gallery
reQuickAccessToolbar	Quick Access Toolbar
reScrollBar	Scroll Bar
reStatusBar	Status Bar

69.2.2 RibbonStyleActnCtrls.TSkinApplicationMenu Enumeration

Individual skinned items of Application Menu.

Pascal

```
  TSkinApplicationMenu = (
    samButton,
    samButtonHover,
    samButtonPressed,
    samButtonShadow,
    samBorder,
    samMenuNormal,
    samMenuDisabled,
    samMenuHover,
```

```
    samMenuPressed,
    samMenuSplitNormal,
    samMenuSplitHover,
    samMenuSplitPressed,
    samMenuSubMenu
  );
```

C++

```
  enum TSkinApplicationMenu {
    samButton,
    samButtonHover,
    samButtonPressed,
    samButtonShadow,
    samBorder,
    samMenuNormal,
    samMenuDisabled,
    samMenuHover,
    samMenuPressed,
    samMenuSplitNormal,
    samMenuSplitHover,
    samMenuSplitPressed,
    samMenuSubMenu
  };
```

File

RibbonStyleActnCtrls

Description

TSkinApplicationMenu enumerates the individual skinned items of the Application Menu.

Value	Meaning
samButton	Button
samButtonHover	Button hover
samButtonPressed	Button pressed
samButtonShadow	Button shadow
samBorder	Border
samMenuNormal	Normal menu
samMenuDisabled	Disabled menu
samMenuHover	Menu hover
samMenuPressed	Pressed split menu
samMenuSplitNormal	Normal split menu
samMenuSplitHover	Split menu hover
samMenuSplitPressed	Pressed split menu
samMenuSubMenu	Submenu menu
samBorderTopStart	Top border start
samBorderTop	Top border
samBorderTopEnd	Top border end
samBorderBottomStart	Bottom border start
samBorderBottom	Bottom border
samBorderBottomEnd	Bottom border end

See Also

TRibbonElement (see page 4029)

69.2.3 RibbonStyleActnCtrls.TSkinButton Enumeration

Individual skinned items of Button.

Pascal

```
TSkinButton = (
  sbSmall,
  sbSmallDisabled,
  sbSmallHover,
  sbSmallPressed,
  sbSmallSelected,
  sbSmallHoverSelected,
  sbSmallInactive,
  sbLarge,
  sbLargeDisabled,
  sbLargeSplitActive,
  sbLargeHover,
  sbLargePressed,
  sbLargeSelected,
  sbLargeHoverSelected,
  sbSmallSplit,
  sbSmallSplitDisabled,
  sbSmallSplitHover,
  sbSmallSplitPressed,
  sbSmallSplitSelected,
  sbSmallSplitHoverSelected,
  sbSmallSplitInactive,
  sbSmallSplitHoverSplit,
  sbSmallSplitPressedSplit,
  sbSmallSplitInactiveSplit,
  sbLargeSplit,
  sbLargeSplitDisabled,
  sbLargeSplitHover,
  sbLargeSplitPressed,
  sbLargeSplitSelected,
  sbLargeSplitHoverSelected,
  sbLargeSplitHoverSplit,
  sbLargeSplitPressedSplit,
  sbArrow,
  sbArrowDisabled,
  sbArrowPressed,
  sbLargeArrowDown,
  sbLargeArrowDownDisabled,
  sbSeparator,
  sbDisabledSeparator,
  sbGroup,
  sbGroupStart,
  sbGroupStartHover,
  sbGroupStartPressed,
  sbGroupStartSelected,
  sbGroupStartHoverSelected,
  sbGroupStartInactive,
  sbGroupMiddle,
  sbGroupMiddleHover,
  sbGroupMiddlePressed,
  sbGroupMiddleSelected,
  sbGroupMiddleHoverSelected,
```

```
    sbGroupMiddleInactive,
    sbGroupEndSplitInactive,
    sbGroupStartSplitInactive,
    sbGroupSingleSplitInactive,
    sbGroupMiddleSplitInactive,
    sbGroupEnd,
    sbGroupEndHover,
    sbGroupEndPressed,
    sbGroupEndSelected,
    sbGroupEndHoverSelected,
    sbGroupEndInactive,
    sbGroupSingle,
    sbGroupSingleHover,
    sbGroupSinglePressed,
    sbGroupSingleSelected,
    sbGroupSingleHoverSelected,
    sbGroupSingleInactive,
    sbGroupSplit,
    sbGroupStartSplitHover,
    sbGroupMiddleSplitHover,
    sbGroupEndSplitHover,
    sbGroupSingleSplitHover,
    sbGroupStartInactiveSplit,
    sbGroupEndSplitInactiveSplit,
    sbGroupStartSplitHoverSplit,
    sbGroupEndSplitHoverSplit,
    sbGroupSingleSplitHoverSplit,
    sbGroupSingleSplitInactiveSplit,
    sbGroupMiddleSplitPressed,
    sbGroupMiddleSplitInactiveSplit,
    sbGroupMiddleSplitHoverSplit,
    sbGroupStartSplitInactiveSplit,
    sbGroupStartSplitPressed,
    sbGroupEndSplitPressed,
    sbGroupSingleSplitPressed,
    sbGroupStartSplitPressedSplit,
    sbGroupMiddleSplitPressedSplit,
    sbGroupEndSplitPressedSplit,
    sbGroupSingleSplitPressedSplit
  );
```

C++

```
  enum TSkinButton {
    sbSmall,
    sbSmallDisabled,
    sbSmallHover,
    sbSmallPressed,
    sbSmallSelected,
    sbSmallHoverSelected,
    sbSmallInactive,
    sbLarge,
    sbLargeDisabled,
    sbLargeSplitActive,
    sbLargeHover,
    sbLargePressed,
    sbLargeSelected,
    sbLargeHoverSelected,
    sbSmallSplit,
    sbSmallSplitDisabled,
    sbSmallSplitHover,
    sbSmallSplitPressed,
    sbSmallSplitSelected,
    sbSmallSplitHoverSelected,
    sbSmallSplitInactive,
    sbSmallSplitHoverSplit,
    sbSmallSplitPressedSplit,
```

```
    sbSmallSplitInactiveSplit,
    sbLargeSplit,
    sbLargeSplitDisabled,
    sbLargeSplitHover,
    sbLargeSplitPressed,
    sbLargeSplitSelected,
    sbLargeSplitHoverSelected,
    sbLargeSplitHoverSplit,
    sbLargeSplitPressedSplit,
    sbArrow,
    sbArrowDisabled,
    sbArrowPressed,
    sbLargeArrowDown,
    sbLargeArrowDownDisabled,
    sbSeparator,
    sbDisabledSeparator,
    sbGroup,
    sbGroupStart,
    sbGroupStartHover,
    sbGroupStartPressed,
    sbGroupStartSelected,
    sbGroupStartHoverSelected,
    sbGroupStartInactive,
    sbGroupMiddle,
    sbGroupMiddleHover,
    sbGroupMiddlePressed,
    sbGroupMiddleSelected,
    sbGroupMiddleHoverSelected,
    sbGroupMiddleInactive,
    sbGroupEndSplitInactive,
    sbGroupStartSplitInactive,
    sbGroupSingleSplitInactive,
    sbGroupMiddleSplitInactive,
    sbGroupEnd,
    sbGroupEndHover,
    sbGroupEndPressed,
    sbGroupEndSelected,
    sbGroupEndHoverSelected,
    sbGroupEndInactive,
    sbGroupSingle,
    sbGroupSingleHover,
    sbGroupSinglePressed,
    sbGroupSingleSelected,
    sbGroupSingleHoverSelected,
    sbGroupSingleInactive,
    sbGroupSplit,
    sbGroupStartSplitHover,
    sbGroupMiddleSplitHover,
    sbGroupEndSplitHover,
    sbGroupSingleSplitHover,
    sbGroupStartInactiveSplit,
    sbGroupEndSplitInactiveSplit,
    sbGroupStartSplitHoverSplit,
    sbGroupEndSplitHoverSplit,
    sbGroupSingleSplitHoverSplit,
    sbGroupSingleSplitInactiveSplit,
    sbGroupMiddleSplitPressed,
    sbGroupMiddleSplitInactiveSplit,
    sbGroupMiddleSplitHoverSplit,
    sbGroupStartSplitInactiveSplit,
    sbGroupStartSplitPressed,
    sbGroupEndSplitPressed,
    sbGroupSingleSplitPressed,
    sbGroupStartSplitPressedSplit,
    sbGroupMiddleSplitPressedSplit,
    sbGroupEndSplitPressedSplit,
```

```
  sbGroupSingleSplitPressedSplit
};
```

File

RibbonStyleActnCtrls

Description

TSkinButton enumerates the individual skinned items of a Button.

Value	Meaning
sbSmall	Small
sbSmallDisabled	Disabled small
sbSmallHover	Small hover
sbSmallPressed	Pressed small
sbSmallSelected	Selected small
sbSmallHoverSelected	Selected small hover
sbSmallGroup	Small group
sbSmallGroupHover	Small group hover
sbSmallGroupPressed	Pressed small group
sbSmallGroupSelected	Selected small group
sbSmallGroupHoverSelected	Selected small group hover
sbLarge	Large
sbLargeDisabled	Disabled Large
sbLargeSplitActive	Active large split
sbLargeHover	Large hover
sbLargePressed	Pressed Large
sbLargeSelected	Selected Large
sbLargeHoverSelected	Selected Large hover
sbSmallSplit	Small split
sbSmallSplitDisabled	Disabled small split
sbSmallSplitHover	Small split hover
sbSmallSplitPressed	Pressed small split
sbSmallSplitSelected	Selected small split
sbSmallSplitHoverSelected	Selected small split hover
sbSmallSplitHoverSplit	Split small split hover
sbSmallSplitPressedSplit	Pressed split small split
sbLargeSplit	Large split
sbLargeSplitDisabled	Disabled large split
sbLargeSplitHover	Large split hover
sbLargeSplitPressed	Pressed large split
sbLargeSplitSelected	Selected large split
sbLargeSplitHoverSelected	Selected large split hover

sbLargeSplitHoverSplit	Split large split hover
sbLargeSplitPressedSplit	Pressed split large split
sbGroup	Group
sbGroupStart	Group start
sbGroupStartHover	Group start hover
sbGroupStartPressed	Pressed group start
sbGroupStartSelected	Selected group start
sbGroupStartHoverSelected	Selected group start hover
sbGroupStartInactive	Inactive group start
sbGroupMiddle	Group middle
sbGroupMiddleHover	Group middle hover
sbGroupMiddlePressed	Pressed group middle
sbGroupMiddleSelected	Selected group middle
sbGroupMiddleHoverSelected	Selected group middle hover
sbGroupMiddleInactive	Inactive group middle
sbGroupEnd	Group end
sbGroupEndHover	Group end hover
sbGroupEndPressed	Pressed group end
sbGroupEndSelected	Selected group end
sbGroupEndHoverSelected	Selected group end hover
sbGroupEndInactive	Inactive group end
sbGroupSingle	Single group
sbGroupSingleHover	Single group hover
sbGroupSinglePressed	Pressed single group
sbGroupSingleSelected	Selected single group
sbGroupSingleHoverSelected	Selected single group hover
sbGroupSingleInactive	Inactive single group

See Also

TRibbonElement (see page 4029)

69.2.4 RibbonStyleActnCtrls.TSkinCheckBox Enumeration

Pascal

```
TSkinCheckBox = (
  scNormal,
  scDisabled,
  scHover,
  scPressed,
  scSelected,
  scHoverSelected,
```

```
  scDisabledSelected
);
```

C++

```
enum TSkinCheckBox {
  scNormal,
  scDisabled,
  scHover,
  scPressed,
  scSelected,
  scHoverSelected,
  scDisabledSelected
};
```

File

RibbonStyleActnCtrls

Description

This is record RibbonStyleActnCtrls.TSkinCheckBox.

69.2.5 RibbonStyleActnCtrls.TSkinComboBox Enumeration

Individual skinned items of Combo.

Pascal

```
TSkinComboBox = (
  scButton,
  scButtonInactive,
  scButtonHover,
  scButtonPressed,
  scButtonIcon
);
```

C++

```
enum TSkinComboBox {
  scButton,
  scButtonInactive,
  scButtonHover,
  scButtonPressed,
  scButtonIcon
};
```

File

RibbonStyleActnCtrls

Description

TSkinComboBox enumerates the individual skinned items of a Combo.

Value	Meaning
scButton	Button
scButtonInactive	Inactive button
scButtonHover	Button hover
scButtonPressed	Pressed button
scButtonIcon	Button icon

See Also

TRibbonElement (see page 4029)

69.2.6 RibbonStyleActnCtrls.TSkinForm Enumeration

Individual skinned items of Form.

Pascal

```
TSkinForm = (
  sfLeftFrame,
  sfLeftFrameDisabled,
  sfLeftFrameTop,
  sfLeftFrameTopDisabled,
  sfRightFrame,
  sfRightFrameDisabled,
  sfRightFrameTop,
  sfRightFrameTopDisabled,
  sfBottomFrame,
  sfBottomFrameDisabled,
  sfButton,
  sfButtonHover,
  sfButtonPressed,
  sfNone,
  sfMinimize,
  sfMinimizeHover,
  sfMinimizePressed,
  sfMinimizeDisabled,
  sfMaximize,
  sfMaximizeHover,
  sfMaximizePressed,
  sfMaximizeDisabled,
  sfRestore,
  sfRestoreHover,
  sfRestorePressed,
  sfRestoreDisabled,
  sfClose,
  sfCloseHover,
  sfClosePressed,
  sfCloseDisabled,
  sfHelp,
  sfHelpHover,
  sfHelpPressed,
  sfHelpDisabled
);
```

C++

```
enum TSkinForm {
  sfLeftFrame,
  sfLeftFrameDisabled,
  sfLeftFrameTop,
  sfLeftFrameTopDisabled,
  sfRightFrame,
  sfRightFrameDisabled,
  sfRightFrameTop,
  sfRightFrameTopDisabled,
  sfBottomFrame,
  sfBottomFrameDisabled,
  sfButton,
  sfButtonHover,
  sfButtonPressed,
```

```
    sfNone,
    sfMinimize,
    sfMinimizeHover,
    sfMinimizePressed,
    sfMinimizeDisabled,
    sfMaximize,
    sfMaximizeHover,
    sfMaximizePressed,
    sfMaximizeDisabled,
    sfRestore,
    sfRestoreHover,
    sfRestorePressed,
    sfRestoreDisabled,
    sfClose,
    sfCloseHover,
    sfClosePressed,
    sfCloseDisabled,
    sfHelp,
    sfHelpHover,
    sfHelpPressed,
    sfHelpDisabled
  };
```

File

RibbonStyleActnCtrls

Description

TSkinForm enumerates the individual skinned items of the Form.

Value	Meaning
sfLeftFrame	Left frame
sfLeftFrameTop	Left frame top
sfLeftFrameTopDisabled	Disabled left frame top
sfRightFrame	Right frame
sfRightFrameTop	Right frame top
sfRightFrameTopDisabled	Disabled right frame top
sfBottomFrame	Bottom frame
sfButton	Button
sfButtonHover	Button hover
sfButtonPressed	Button pressed
sfMinimize	Minimize
sfMinimizeHover	Minimize hover
sfMinimizePressed	Minimize pressed
sfMinimizeDisabled	Minimize disabled
sfMaximize	Maximize
sfMaximizeHover	Maximize hover
sfMaximizePressed	Maximize pressed
sfMaximizeDisabled	Maximize disabled
sfRestore	Restore
sfRestoreHover	Restore hover

sfRestorePressed	Restore pressed
sfRestoreDisabled	Restore disabled
sfClose	Close
sfCloseHover	Close hover
sfClosePressed	Close pressed
sfCloseDisabled	Close disabled

See Also

TRibbonElement (see page 4029)

69.2.7 RibbonStyleActnCtrls.TSkinGallery Enumeration

Individual skinned items of Gallery.

Pascal

```
TSkinGallery = (
  sgButton,
  sgUpButton,
  sgUpButtonHover,
  sgUpButtonPressed,
  sgUpButtonDisabled,
  sgDownButton,
  sgDownButtonHover,
  sgDownButtonPressed,
  sgDownButtonDisabled,
  sgMoreButton,
  sgMoreButtonHover,
  sgMoreButtonHoverDisabled
);
```

C++

```
enum TSkinGallery {
  sgButton,
  sgUpButton,
  sgUpButtonHover,
  sgUpButtonPressed,
  sgUpButtonDisabled,
  sgDownButton,
  sgDownButtonHover,
  sgDownButtonPressed,
  sgDownButtonDisabled,
  sgMoreButton,
  sgMoreButtonHover,
  sgMoreButtonHoverDisabled
};
```

File

RibbonStyleActnCtrls

Description

TSkinGallery enumerates the individual skinned items of a Gallery.

Value	Meaning
sgBackground	Background
sgUpButton	Up button
sgUpButtonHover	Up button hover
sgUpButtonPressed	Pressed up button
sgupButtonDisabled	Disabled up button
sgDownButton	Down button
sgDownButtonHover	Down button hover
sgDownButtonPressed	Pressed down button
sgDownButtonDisabled	Disabled down button
sgMoreButton	More button
sgMoreButtonHover	More button hover
sgUpIcon	Up icon
sgDownIcon	Down icon
sgMoreIcon	More icon

See Also

TRibbonElement (see page 4029)

69.2.8 RibbonStyleActnCtrls.TSkinMenu Enumeration

Individual skinned items of Menu.

Pascal

```
TSkinMenu = (
  smBackground,
  smBanner,
  smNormal,
  smSelected,
  smHover,
  smDisabled,
  smCheck,
  smCheckDisabled,
  smCheckIcon,
  smCheckDisabledIcon,
  smRadio,
  smRadioDisabled,
  smSeparator,
  smArrowLeftIcon,
  smArrowLeftDisabledIcon,
  smArrowRightIcon,
  smArrowRightDisabledIcon,
  smResize,
  smResizeVertical,
  smResizeBoth,
  smRichContentHover,
  smRichContentDisabled,
  smScrollButton,
  smScrollButtonHover,
```

```
    smScrollButtonUpIcon,
    smScrollButtonDownIcon,
    smOptionButton,
    smOptionButtonHover,
    smOptionButtonDisabled
  );
```

C++

```
  enum TSkinMenu {
    smBackground,
    smBanner,
    smNormal,
    smSelected,
    smHover,
    smDisabled,
    smCheck,
    smCheckDisabled,
    smCheckIcon,
    smCheckDisabledIcon,
    smRadio,
    smRadioDisabled,
    smSeparator,
    smArrowLeftIcon,
    smArrowLeftDisabledIcon,
    smArrowRightIcon,
    smArrowRightDisabledIcon,
    smResize,
    smResizeVertical,
    smResizeBoth,
    smRichContentHover,
    smRichContentDisabled,
    smScrollButton,
    smScrollButtonHover,
    smScrollButtonUpIcon,
    smScrollButtonDownIcon,
    smOptionButton,
    smOptionButtonHover,
    smOptionButtonDisabled
  };
```

File

RibbonStyleActnCtrls

Description

TSkinMenu enumerates the individual skinned items of a Menu.

Value	Meaning
smBackground	Background
smBanner	Banner
smNormal	Normal
smSelected	Selected
smSelectedStart	Selected start
smSelectedEnd	Selected end
smHover	Hover
smHoverStart	Hover start
smHoverEnd	Hover end
smHoverSelected	Selected hover

smDisabled	Disabled
smCheck	Check
smCheckDisabled	Disabled check
smCheckIcon	Check icon
smCheckDisabledIcon	Disabled check icon
smRadio	Radio
smRadioDisabled	Disabled radio
smSeparator	Separator
smArrowLeftIcon	Left arrow icon
smArrowRightIcon	Right arrow icon
smResize	Resize
smResizeVertical	Vertical resize
smResizeBoth	Resize both
smRichContentHover	Rich content hover
smRichContentDisabled	Disabled rich content
smScrollButton	Scroll button
smScrollButtonHover	Scroll button hover
smScrollButtonUpIcon	Up scroll button icon
smScrollButtonDownIcon	Down scroll button icon
smOptionButton	Option button
smOptionButtonHover	Option button hover
smOptionButtonDisabled	Disabled option button

See Also

TRibbonElement (see page 4029)

69.2.9 RibbonStyleActnCtrls.TSkinQuickAccessToolbar Enumeration

Individual skinned items of Quick Access Toolbar.

Pascal

```
TSkinQuickAccessToolbar = (
  sqBackground,
  sqStart,
  sqStartRTL,
  sqMiddle,
  sqEnd,
  sqEndRTL,
  sqButton,
  sqButtonHover,
  sqButtonPressed,
  sqButtonCustomizeIcon,
```

```
    sqButtonMoreIcon,
    sqGroup,
    sqGroupHover,
    sqGroupPressed
  );
```

C++

```
  enum TSkinQuickAccessToolbar {
    sqBackground,
    sqStart,
    sqStartRTL,
    sqMiddle,
    sqEnd,
    sqEndRTL,
    sqButton,
    sqButtonHover,
    sqButtonPressed,
    sqButtonCustomizeIcon,
    sqButtonMoreIcon,
    sqGroup,
    sqGroupHover,
    sqGroupPressed
  };
```

File

RibbonStyleActnCtrls

Description

TSkinQuickAccessToolbar enumerates the individual skinned items of the Quick Access Toolbar.

Value	Meaning
sqBackground	Background
sqStart	Start
sqMiddle	Middle
sqEnd	End
sqButton	Button
sqButtonHover	Button hover
sqButtonPressed	Pressed button
sqButtonCustomizeIcon	Customize button icon
sqButtonMoreIcon	More button icon
sqGroup	Group
sqGroupHover	Group hover
sqGroupPressed	Pressed group

See Also

TRibbonElement (see page 4029)

69.2.10 RibbonStyleActnCtrls.TSkinRadioButton Enumeration

Pascal

```
TSkinRadioButton = (
  sbNormal,
  sbDisabled,
  sbHover,
  sbPressed,
  sbSelected,
  sbHoverSelected,
  sbDisabledSelected
);
```

C++

```
enum TSkinRadioButton {
  sbNormal,
  sbDisabled,
  sbHover,
  sbPressed,
  sbSelected,
  sbHoverSelected,
  sbDisabledSelected
};
```

File

RibbonStyleActnCtrls

Description

This is record RibbonStyleActnCtrls.TSkinRadioButton.

69.2.11 RibbonStyleActnCtrls.TSkinRibbon Enumeration

Individual skinned items of Ribbon (see page 3347).

Pascal

```
TSkinRibbon = (
  srBackground,
  srBackgroundDisabled,
  srHeader,
  srBody,
  srPage,
  srHelp
);
```

C++

```
enum TSkinRibbon {
  srBackground,
  srBackgroundDisabled,
  srHeader,
  srBody,
  srPage,
  srHelp
```

```
};
```

File

RibbonStyleActnCtrls

Description

TSkinRibbon enumerates the individual skinned items of the Ribbon (see page 3347).

Value	Meaning
srBackground	Background
srBackgroundDisabled	Disabled background
srHeader	Header
srBody	Body
srPageStart	Page start
srPage	Page
srPageEnd	Page end
srHelp	Help

See Also

TRibbonElement (see page 4029)

69.2.12 RibbonStyleActnCtrls.TSkinRibbonGroup Enumeration

Individual skinned items of Group.

Pascal

```
TSkinRibbonGroup = (
  srgBackground,
  srgBackgroundHover,
  srgBackgroundDisabled,
  srgCaption,
  srgCaptionHover,
  srgCaptionDisabled,
  srgDialogButton,
  srgDialogButtonHover,
  srgDialogButtonPressed,
  srgDialogButtonIcon,
  srgDialogButtonIconHover,
  srgDialogButtonIconPressed,
  srgMinimizeButton,
  srgMinimizeButtonHover,
  srgMinimizeButtonPressed,
  srgMinimizeButtonIcon,
  srgMinimizeButtonPressedIcon
);
```

C++

```
enum TSkinRibbonGroup {
  srgBackground,
  srgBackgroundHover,
  srgBackgroundDisabled,
```

```
    srgCaption,
    srgCaptionHover,
    srgCaptionDisabled,
    srgDialogButton,
    srgDialogButtonHover,
    srgDialogButtonPressed,
    srgDialogButtonIcon,
    srgDialogButtonIconHover,
    srgDialogButtonIconPressed,
    srgMinimizeButton,
    srgMinimizeButtonHover,
    srgMinimizeButtonPressed,
    srgMinimizeButtonIcon,
    srgMinimizeButtonPressedIcon
  };
```

File

RibbonStyleActnCtrls

Description

TSkinRibbonGroup enumerates the individual skinned items of a Group.

Value	Meaning
srgBackground	Background
srgBackgroundStart	Background start
srgBackgroundEnd	Background end
srgBackgroundHover	Background hover
srgBackgroundHoverStart	Background start hover
srgBackgroundHoverEnd	Background end hover
srgBackgroundDisabled	Disabled background
srgBackgroundDisabledStart	Disabled background start
srgBackgroundDisabledEnd	Disabled background end
srgCaption	Caption
srgCaptionHover	Caption hover
srgCaptionDisabled	Disabled caption
srgDialogButton	Dialog button
srgDialogButtonHover	Dialog button hover
srgDialogButtonPressed	Pressed dialog button
srgDialogButtonIcon	Dialog button icon
srgDialogButtonIconHover	Dialog button icon hover
srgDialogButtonIconPressed	Pressed dialog button icon
srgMinimizeButton	Minimize button
srgMinimizeButtonStart	Minimize button start
srgMinimizeButtonEnd	Minimize button end
srgMinimizeButtonHover	Minimize button hover
srgMinimizeButtonHoverStart	Minimize button hover start
srgMinimizeButtonHoverEnd	Minimize button hover end

srgMinimizeButtonPressed	Pressed minimize button
srgMinimizeButtonPressedStart	Pressed minimize button start
srgMinimizeButtonPressedEnd	Pressed minimize button end
srgMinimizeButtonIcon	Minimize button icon
srgMinimizeButtonPressedIcon	Pressed minimize button icon
srgScrollLeft	Left scroll
srgScrollLeftHover	Left scroll hover
srgScrollRight	Scroll right
srgScrollRightHover	Scroll right hover

See Also

TRibbonElement (see page 4029)

69.2.13 RibbonStyleActnCtrls.TSkinScrollBar Enumeration

Individual skinned items of ScrollBar.

Pascal

```
TSkinScrollBar = (
  ssbBar,
  ssbBarHover,
  ssbBarPressed,
  ssbBarIcon,
  ssbButton,
  ssbButtonActive,
  ssbButtonDisabled,
  ssbButtonHover,
  ssbButtonPressed,
  ssbButtonUpIcon,
  ssbButtonDownIcon
);
```

C++

```
enum TSkinScrollBar {
  ssbBar,
  ssbBarHover,
  ssbBarPressed,
  ssbBarIcon,
  ssbButton,
  ssbButtonActive,
  ssbButtonDisabled,
  ssbButtonHover,
  ssbButtonPressed,
  ssbButtonUpIcon,
  ssbButtonDownIcon
};
```

File

RibbonStyleActnCtrls

Description

TSkinScrollBar enumerates the individual skinned items of a ScrollBar.

Value	Meaning
ssbBar	Bar
ssbBarHover	Bar hover
ssbBarPressed	Pressed bar
ssbBarIcon	Bar icon
ssbButton	Button
ssbButtonActive	Active button
ssbButtonDisabled	Disabled button
ssbButtonHover	Button hover
ssbButtonPressed	Pressed button
ssbButtonUpIcon	Button up icon
ssbButtonDownIcon	Button down icon

See Also

TRibbonElement (see page 4029)

69.2.14 RibbonStyleActnCtrls.TSkinSpinEdit Enumeration

Individual skinned items of Spinner.

Pascal

```
TSkinSpinEdit = (
  ssButton,
  ssButtonDisabled,
  ssButtonHover,
  ssButtonInactive,
  ssButtonPressed,
  ssButtonUp,
  ssButtonDown,
  ssButtonUpIcon,
  ssButtonDownIcon
);
```

C++

```
enum TSkinSpinEdit {
  ssButton,
  ssButtonDisabled,
  ssButtonHover,
  ssButtonInactive,
  ssButtonPressed,
  ssButtonUp,
  ssButtonDown,
  ssButtonUpIcon,
  ssButtonDownIcon
};
```

File

RibbonStyleActnCtrls

Description

TSkinSpinEdit enumerates the individual skinned items of a Spinner.

Value	Meaning
ssButton	Button
ssButtonDisabled	Disabled button
ssButtonHover	Button hover
ssButtonInactive	Inactive button
ssButtonPressed	Pressed button
ssButtonUp	Button up
ssButtonDown	Button down
ssButtonUpIcon	Button up icon
ssButtonDownIcon	Button down icon

See Also

TRibbonElement (see page 4029)

69.2.15 RibbonStyleActnCtrls.TSkinStatusBar Enumeration

Individual skinned items of Status Bar.

Pascal

```
TSkinStatusBar = (
  ssbrBackground,
  ssbrNotification,
  ssbrControl,
  ssbrGripper
);
```

C++

```
enum TSkinStatusBar {
  ssbrBackground,
  ssbrNotification,
  ssbrControl,
  ssbrGripper
};
```

File

RibbonStyleActnCtrls

Description

TSkinStatusBar enumerates the individual skinned items of a Status Bar.

Value	Meaning
ssbrBackground	Background
ssbrNotification	Notification
ssbrControl	Control

ssbrGripper	Gripper

See Also

TRibbonElement (see page 4029)

69.2.16 RibbonStyleActnCtrls.TSkinTab Enumeration

Individual skinned items of Tab.

Pascal

```
TSkinTab = (
  stBackground,
  stNormal,
  stHover,
  stHoverUnselected,
  stHoverSelected,
  stKeyTipsActive,
  stScrollLeft,
  stScrollLeftHover,
  stScrollLeftPressed,
  stScrollRight,
  stScrollRightHover,
  stScrollRightPressed,
  stSeparator,
  stScrollGroupLeft,
  stScrollGroupLeftHover,
  stScrollGroupRight,
  stScrollGroupRightHover
);
```

C++

```
enum TSkinTab {
  stBackground,
  stNormal,
  stHover,
  stHoverUnselected,
  stHoverSelected,
  stKeyTipsActive,
  stScrollLeft,
  stScrollLeftHover,
  stScrollLeftPressed,
  stScrollRight,
  stScrollRightHover,
  stScrollRightPressed,
  stSeparator,
  stScrollGroupLeft,
  stScrollGroupLeftHover,
  stScrollGroupRight,
  stScrollGroupRightHover
};
```

File

RibbonStyleActnCtrls

Description

TSkinTab enumerates the individual skinned items of a Tab.

Value	Meaning
stBackground	Background
stHoverUnselected	Unselected hover
stHoverUnselectedStart	Unselected start hover
stHoverUnselectedEnd	Unselected end hover
stHover	Hover
stHoverStart	Start hover
stHoverEnd	End hover
stNormal	Normal
stNormalStart	Start normal
stNormalEnd	End normal
stHoverSelected	Selected hover
stScrollLeft	Left scroll
stScrollLeftPressed	Pressed Left scroll
stScrollLeftHover	Left scroll hover
stScrollRight	Right scroll
stScrollRightPressed	Pressed right scroll
stScrollRightHover	Right scroll hover
stKeyTipsActive	KeyTips active
stKeyTipsActiveStart	KeyTips active start
stKeyTipsActiveEnd	KeyTips active end
stSeparator	Separator

See Also

TRibbonElement (see page 4029)

69.3 Types

The following table lists types in this documentation.

Types

Name	Description
TCustomRibbonSkinClass (see page 4053)	This is type RibbonStyleActnCtrls.TCustomRibbonSkinClass.

69.3.1 RibbonStyleActnCtrls.TCustomRibbonSkinClass Type

Pascal

```
TCustomRibbonSkinClass = class of TCustomRibbonSkin;
```

C++

```
TCustomRibbonSkin TCustomRibbonSkinClass;
```

File

RibbonStyleActnCtrls

Description

This is type RibbonStyleActnCtrls.TCustomRibbonSkinClass.

70 Rio Namespace

This is namespace Rio.

Topics

Name	Description
Classes (see page 4056)	The following table lists classes in this documentation.
Interfaces (see page 4068)	The following table lists interfaces in this documentation.
Types (see page 4070)	The following table lists types in this documentation.
Constants (see page 4072)	The following table lists constants in this documentation.

Classes

	Name	Description
	TRIO (see page 4056)	TRIO is the base class for components that generate calls to remote interfaced objects.

Constants

Name	Description
StubSize (see page 4072)	This is constant Rio.StubSize.

Interfaces

	Name	Description
	IRIOAccess (see page 4068)	IRIOAccess is the interface for accessing a remote interfaced object from its invokable interface.

Types

Name	Description
TAfterExecuteEvent (see page 4070)	TAfterExecuteEvent is the type of event handlers that respond after a remote interfaced object receives the encoded results from executing a method call to a Web Service.
TBeforeExecuteEvent (see page 4070)	TBeforeExecuteEvent is the type of event handlers that respond before a remote interfaced object sends an encoded method call to a Web Service provider.

70.1 Classes

The following table lists classes in this documentation.

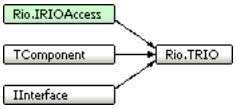
Classes

	Name	Description
	TRIO (see page 4056)	TRIO is the base class for components that generate calls to remote interfaced objects.

70.1.1 TRIO Class

TRIO is the base class for components that generate calls to remote interfaced objects.

Class Hierarchy



Pascal

```
TRIO = class(TComponent, IInterface, IRIOAccess);
```

C++

```
class TRIO : public TComponent, public IInterface, public IRIOAccess;
```

File

Rio

Description

Use TRIO as a base class for components that generate statically-linked calls to invokable interfaces. When an application casts a TRIO descendant to a registered invokable interface, it dynamically generates an in-memory method table, providing an implementation to that invokable interface. TRIO uses two helper objects when generating this implementation. These helper objects implement the IOPConvert (see page 2730) and IWebNode interfaces. IOPConvert (see page 2730) handles the marshaling and unmarshaling of calls to methods on the generated method table, and IWebNode receives the marshaled message call, executes it (typically by sending it to a remote server application), and returns the results of executing the call. These interfaces do not assume any particular encoding or transport protocol. Each TRIO descendant uses its own protocol-specific helper objects to implement these interfaces.

Before TRIO can generate a method table for an invokable interface, the interface must be registered with the invocation registry. To register an invokable interface, use the global InvRegistry (see page 930) function to obtain access to the registry, and call its RegisterInterface method.

Do not create instances of TRIO. To add a component that generates calls to a registered invokable interface, use a TRIO descendant such as THTTPRIO.

See Also

THTTPRIO

InvRegistry (see page 930)

TInvokableClassRegistry (see page 840)

IWebNode

IOPConvert (see page 2730)

IRIOAccess (see page 4068)

70.1.1.1 TRIO Members

The following tables list the members exposed by TRIO.

Protected Fields

	Name	Description
	FConverter (see page 4059)	This is FConverter, a member of class TRIO.
	FIID (see page 4059)	This is FIID, a member of class TRIO.
	FWebNode (see page 4059)	This is FWebNode, a member of class TRIO.
	IntfMD (see page 4059)	This is IntfMD, a member of class TRIO.

Protected Methods

	Name	Description
V	DoAfterExecute (see page 4061)	Generates an OnAfterExecute (see page 4065) event.
V	DoBeforeExecute (see page 4061)	Generates an OnBeforeExecute (see page 4066) event.
V	GetResponseStream (see page 4062)	Returns a newly created stream that receives the results of executing a method call.
V	QueryInterface (see page 4063)	Implements the IInterface QueryInterface method.

Public Methods

	Name	Description
V	AfterConstruction (see page 4060)	Updates the reference count after the last constructor has executed.
V	BeforeDestruction (see page 4060)	Prevents the destruction of a TRIO (see page 4056) instance if the reference count is greater than zero.
V	Create (see page 4058)	Creates an instance of TRIO (see page 4056).
V	Destroy (see page 4058)	Destroys the instance of a TRIO (see page 4056) object.
	GetRIO (see page 4069)	Returns the value of the RIO (see page 4069) property.
V	NewInstance (see page 4062)	Allocates memory for each instance of a remote interfaced object.

Public Properties

	Name	Description
	Converter (see page 4063)	Handles the marshaling and unmarshaling of calls to an invokable interface.
R	RefCount (see page 4064)	Maintains a reference count on the use of the TRIO (see page 4056) object's interface.
R	RIO (see page 4069)	Provides access to the remote interfaced object that implements this interface.
R	SOAPHeaders (see page 4064)	Provides access to the headers of SOAP messages that the remote interface object sends or receives.
	WebNode (see page 4065)	Handles the execution of method calls on the invokable interface.

Published Events

	Name	Description
	OnAfterExecute (see page 4065)	Occurs when TRIO (see page 4056) receives the results from executing a method of the invokable interface.
	OnBeforeExecute (see page 4066)	Occurs before TRIO (see page 4056) sends an encoded method call for execution.
	OnGetAttachment (see page 4067)	Occurs when TRIO (see page 4056) reads each attachment included in the results from executing a method of the invokable interface.
	OnSendAttachment (see page 4067)	Occurs when TRIO (see page 4056) sends each attachment that is the value of an argument of the method call on the invokable interface.

70.1.1.2 TRIO.Create Constructor

Creates an instance of TRIO (see page 4056).

Pascal

```
constructor Create(AOwner: TComponent); override;
```

C++

```
virtual __fastcall TRIO(TComponent * AOwner);
```

Description

Call Create to instantiate a TRIO (see page 4056) descendant at runtime. TRIO (see page 4056) descendants added to forms or data modules at design time are created automatically.

The AOwner parameter specifies another component (usually a form or data module) that is responsible for freeing the TRIO (see page 4056) instance. It becomes the value of the Owner property. If the TRIO (see page 4056) instance is created with AOwner set to nil (Delphi) or NULL (C++), then it is automatically freed when its reference count drops to zero.

See Also

RefCount (see page 4064)

Owner

70.1.1.3 TRIO.Destroy Destructor

Destroys the instance of a TRIO (see page 4056) object.

Pascal

```
destructor Destroy; override;
```

C++

```
virtual __fastcall ~TRIO();
```

Description

There is no need to explicitly free a TRIO (see page 4056) instance.

If the TRIO (see page 4056) object is created with an Owner, then that Owner handles the freeing of the TRIO (see page 4056) instance. If the TRIO (see page 4056) object is created without an Owner, then it is automatically freed when the reference count on its interface drops to zero.

See Also

Create (see page 4058)

Owner

RefCount (see page 4064)

70.1.1.4 TRIO Fields

The fields of the TRIO class are listed here.

Protected Fields

	Name	Description
	FConverter (see page 4059)	This is FConverter, a member of class TRIO.
	FIID (see page 4059)	This is FIID, a member of class TRIO.
	FWebNode (see page 4059)	This is FWebNode, a member of class TRIO.
	IntfMD (see page 4059)	This is IntfMD, a member of class TRIO.

70.1.1.4.1 TRIO.FConverter Field

Pascal

```
FConverter: IOPConvert;
```

C++

```
IOPConvert FConverter;
```

Description

This is FConverter, a member of class TRIO.

70.1.1.4.2 TRIO.FIID Field

Pascal

```
FIID: TGUID;
```

C++

```
TGUID FIID;
```

Description

This is FIID, a member of class TRIO.

70.1.1.4.3 TRIO.FWebNode Field

Pascal

```
FWebNode: IWebNode;
```

C++

```
IWebNode FWebNode;
```

Description

This is FWebNode, a member of class TRIO.

70.1.1.4.4 TRIO.IntfMD Field

Pascal

```
IntfMD: TIntfMetaData;
```

C++

```
TIntfMetaData IntfMD;
```

Description

This is IntfMD, a member of class TRIO.

70.1.1.5 TRIO Methods

The methods of the TRIO class are listed here.

Protected Methods

	Name	Description
	DoAfterExecute (see page 4061)	Generates an OnAfterExecute (see page 4065) event.
	DoBeforeExecute (see page 4061)	Generates an OnBeforeExecute (see page 4066) event.
	GetResponseStream (see page 4062)	Returns a newly created stream that receives the results of executing a method call.
	QueryInterface (see page 4063)	Implements the IInterface QueryInterface method.

Public Methods

	Name	Description
	AfterConstruction (see page 4060)	Updates the reference count after the last constructor has executed.
	BeforeDestruction (see page 4060)	Prevents the destruction of a TRIO (see page 4056) instance if the reference count is greater than zero.
	NewInstance (see page 4062)	Allocates memory for each instance of a remote interfaced object.

70.1.1.5.1 TRIO.AfterConstruction Method

Updates the reference count after the last constructor has executed.

Pascal

```
procedure AfterConstruction; override;
```

C++

```
virtual __fastcall AfterConstruction();
```

Description

AfterConstruction is called automatically after the TRIO (see page 4056) object's last constructor has executed. Do not call it explicitly in your applications.

TRIO (see page 4056) overrides the AfterConstruction method to update the RefCount (see page 4064) property. TRIO (see page 4056) artificially increments the reference count when it is first created to prevent changes to the reference count from causing the TRIO (see page 4056) instance to be freed before it is fully instantiated. AfterConstruction returns the reference count back to an appropriate level, now that the constructor is finished.

See Also

RefCount (see page 4064)

Create (see page 4058)

NewInstance (see page 4062)

70.1.1.5.2 TRIO.BeforeDestruction Method

Prevents the destruction of a TRIO (see page 4056) instance if the reference count is greater than zero.

Pascal

```
procedure BeforeDestruction; override;
```

C++

```
virtual __fastcall BeforeDestruction();
```

Description

BeforeDestruction is called automatically before the TRIO (see page 4056) object's first destructor executes. Do not call it explicitly in your applications.

TRIO (see page 4056) overrides the BeforeDestruction method to check the RefCount (see page 4064) property. If RefCount (see page 4064) is greater than zero, then there is still an outstanding reference to the TRIO (see page 4056) object's interface,. and BeforeDestruction raises an exception to prevent the destruction of this object.

See Also

RefCount (see page 4064)

70.1.1.5.3 TRIO.DoAfterExecute Method

Generates an OnAfterExecute (see page 4065) event.

Pascal

```
procedure DoAfterExecute(const MethodName: string; Response: TStream); virtual;
```

C++

```
virtual __fastcall DoAfterExecute(const UnicodeString MethodName, TStream Response);
```

Description

TRIO (see page 4056) calls DoAfterExecute internally after it receives the results of a method call but before it unmarshals those results. This method simply generates an OnAfterExecute (see page 4065) event, passing the MethodName and Response parameters to the event handler. Override this method to perform class-specific actions before or after the OnAfterExecute (see page 4065) event.

MethodName is the name of the method that was just executed on the server.

Response is a stream from that contains the results that the server returned after executing the specified method.

See Also

OnAfterExecute (see page 4065)

70.1.1.5.4 TRIO.DoBeforeExecute Method

Generates an OnBeforeExecute (see page 4066) event.

Pascal

```
procedure DoBeforeExecute(const MethodName: string; Request: TStream); virtual;
```

C++

```
virtual __fastcall DoBeforeExecute(const UnicodeString MethodName, TStream Request);
```

Description

TRIO (see page 4056) calls DoBeforeExecute internally before it sends a SOAP packet to the server for execution. This method simply generates an OnBeforeExecute (see page 4066) event, passing the MethodName parameter and the string in the Request parameter to the event handler. Override this method to perform class-specific actions before or after the OnBeforeExecute (see page 4066) event.

MethodName is the name of the method that the encoded SOAP request calls.

Request is a stream from which the encoded SOAP request packet can be read.

See Also

OnBeforeExecute (see page 4066)

70.1.1.5.5 TRIO.GetResponseStream Method

Returns a newly created stream that receives the results of executing a method call.

Pascal

```
function GetResponseStream(BindingType: TWebServiceBindingType): TStream; virtual;
```

C++

```
virtual __fastcall TStream GetResponseStream(TWebServiceBindingType BindingType);
```

Description

TRIO (see page 4056) calls GetResponseStream internally when it executes a method call. It passes the returned stream as the Response parameter of the Execute method on the interface that the WebNode (see page 4065) property specifies. After this stream receives the encoded response of a method call, the RIO (see page 4069) component decodes the response and frees the stream.

BindingType indicates the expected coding of the response that the stream will hold. GetResponseStream can use this information to determine what type of stream to create.

See Also

Execute

WebNode (see page 4065)

70.1.1.5.6 TRIO.NewInstance Method

Allocates memory for each instance of a remote interfaced object.

Pascal

```
class function NewInstance: TObject; override;
```

C++

```
virtual __fastcall TObject * NewInstance();
```

Description

All constructors call NewInstance automatically. NewInstance calls InstanceSize to determine how much memory to allocate from the heap to contain a particular instance. Do not call NewInstance directly.

TRIO (see page 4056) overrides NewInstance to increment the reference count when a new instance is created. This prevents any constructor from accidentally deleting the instance due to a RefCount (see page 4064) of zero. The AfterConstruction (see page 4060) method decrements the value of RefCount (see page 4064) after all constructors have executed.

See Also

System::TObject::InstanceSize

System::TObject::NewInstance

AfterConstruction (see page 4060)

70.1.1.5.7 TRIO.QueryInterface Method

Implements the IInterface QueryInterface method.

Pascal

```
function QueryInterface(const IID: TGUID; out Obj): HResult; override; stdcall;
```

C++

```
virtual __fastcall __stdcall HRESULT QueryInterface(const TGUID IID,  Obj);
```

Description

QueryInterface is called through the IInterface interface to obtain an interface pointer for the interface identified by the IID parameter. In Delphi, this call is generated when an application casts TRIO (see page 4056) to an invokable interface using the as operator.

TRIO (see page 4056) can only return a single type of interface pointer. The first time an application calls QueryInterface, TRIO (see page 4056) generates an in-memory method table to support calls to the methods on the requested interface.

IID specifies the invokable interface that the application wants to call using this TRIO (see page 4056) instance. If the TRIO (see page 4056) object has already been typecast to an interface, then IID must identify that interface. If the TRIO (see page 4056) instance has not yet been typecast to an interface, then IID must be a registered invokable interface.

Obj returns an interface pointer for the specified interface, unless TRIO (see page 4056) was already implementing a different interface or IID identified an interface that was not registered.

QueryInterface returns 0 (S_OK) if Obj successfully returns an interface pointer. It returns E_NOINTERFACE otherwise.

If QueryInterface successfully returns an interface pointer, it automatically increments the reference count.

See Also

InvRegistry (see page 930)

RegisterInterface (see page 864)

RefCount (see page 4064)

70.1.1.6 TRIO Properties

The properties of the TRIO class are listed here.

Public Properties

	Name	Description
	Converter (see page 4063)	Handles the marshaling and unmarshaling of calls to an invokable interface.
R	RefCount (see page 4064)	Maintains a reference count on the use of the TRIO (see page 4056) object's interface.
R	SOAPHeaders (see page 4064)	Provides access to the headers of SOAP messages that the remote interface object sends or receives.
	WebNode (see page 4065)	Handles the execution of method calls on the invokable interface.

70.1.1.6.1 TRIO.Converter Property

Handles the marshaling and unmarshaling of calls to an invokable interface.

Pascal

```
property Converter: IOPConvert;
```

C++

```
__property IOPConvert Converter;
```

Description

TRIO (see page 4056) uses Converter internally to marshal and unmarshal calls to an invokable interface. That is, when the TRIO (see page 4056) object has been typecast to an invokable interface, it implements each method call by using Converter to marshal the call into an encoding protocol, and to unmarshal any return values after that call is executed.

Each TRIO (see page 4056) descendant uses its own protocol-specific helper object as the value of the Converter property.

See Also

WebNode (see page 4065)

TOPToSoapDomConvert (see page 2747)

70.1.1.6.2 TRIO.RefCount Property

Maintains a reference count on the use of the TRIO (see page 4056) object's interface.

Pascal

```
property RefCount: Integer;
```

C++

```
__property int RefCount;
```

Description

RefCount keeps track of the usage on the TRIO (see page 4056) object's interface. RefCount is incremented by the interface's _AddRef method (Delphi) or AddRef method (C++) and decremented by its _Release method (Delphi) or Release method (C++).

If the TRIO (see page 4056) object is created without an Owner, then when RefCount drops to zero, the TRIO (see page 4056) instance is automatically freed.

See Also

QueryInterface (see page 4063)

Create (see page 4058)

_AddRef

_Release

70.1.1.6.3 TRIO.SOAPHeaders Property

Provides access to the headers of SOAP messages that the remote interface object sends or receives.

Pascal

```
property SOAPHeaders: TSOAPHeaders;
```

C++

```
__property TSOAPHeaders SOAPHeaders;
```

Description

SOAPHeaders is a helper object that lets TRIO (see page 4056) support the ISOAPHeaders (see page 923) interface. It has methods to add headers to an outgoing request message and to retrieve the headers from the resulting response message. You can use the SOAPHeaders property directly, or access it through an ISOAPHeaders (see page 923) interface, which is

obtained from the invokable interface using the as operator (Delphi) or the Supports method (C++).

Note: Although TRIO (see page 4056) uses the SOAPHeaders property to implement an ISOAPHeaders (see page 923) interface, you can't obtain an ISOAPHeaders (see page 923) interface directly from TRIO (see page 4056). Rather, you must obtain this interface from an invokable interface.

See Also

ISOAPHeaders (see page 923)

70.1.1.6.4 TRIO.WebNode Property

Handles the execution of method calls on the invokable interface.

Pascal

```
property WebNode: IWebNode;
```

C++

```
__property IWebNode WebNode;
```

Description

TRIO (see page 4056) uses WebNode internally to execute method calls after they have been marshaled by Converter (see page 4063). That is, when the TRIO (see page 4056) object has been typecast to an invokable interface, it implements each method call by using Converter (see page 4063) to marshal the call into an encoding protocol, and then uses WebNode to execute the encoded call.

Each TRIO (see page 4056) descendant uses its own protocol-specific helper object as the value of the WebNode property.

See Also

Converter (see page 4063)

THTTPReqResp

OnAfterExecute (see page 4065)

OnBeforeExecute (see page 4066)

70.1.1.7 TRIO Events

The events of the TRIO class are listed here.

Published Events

	Name	Description
	OnAfterExecute (see page 4065)	Occurs when TRIO (see page 4056) receives the results from executing a method of the invokable interface.
	OnBeforeExecute (see page 4066)	Occurs before TRIO (see page 4056) sends an encoded method call for execution.
	OnGetAttachment (see page 4067)	Occurs when TRIO (see page 4056) reads each attachment included in the results from executing a method of the invokable interface.
	OnSendAttachment (see page 4067)	Occurs when TRIO (see page 4056) sends each attachment that is the value of an argument of the method call on the invokable interface.

70.1.1.7.1 TRIO.OnAfterExecute Event

Occurs when TRIO (see page 4056) receives the results from executing a method of the invokable interface.

Pascal

```
property OnAfterExecute: TAfterExecuteEvent;
```

C++

```
__property TAfterExecuteEvent OnAfterExecute;
```

Description

Write an OnAfterExecute event handler to examine or alter the encoded results from executing a method call. OnAfterExecute occurs after TRIO (see page 4056) calls the Execute method of the interface that the WebNode (see page 4065) property specifies, but before the interface that the Converter (see page 4063) property specifies unmarshals those results.

You can use the SOAPResponse stream of the event handler to alter the encoded results from the server before they are unmarshaled. This lets you compensate for known idiosyncrasies of a server or of the Converter (see page 4063) that is the value of the Converter (see page 4063) property. For example, if the Web Service provider publishes interoperability information, you can use this event to adjust SOAP packets to compensate for differences that the Web Service provider reports.

See Also

OnBeforeExecute (see page 4066)

Execute

WebNode (see page 4065)

Converter (see page 4063)

OnGetAttachment (see page 4067)

70.1.1.7.2 TRIO.OnBeforeExecute Event

Occurs before TRIO (see page 4056) sends an encoded method call for execution.

Pascal

```
property OnBeforeExecute: TBeforeExecuteEvent;
```

C++

```
__property TBeforeExecuteEvent OnBeforeExecute;
```

Description

Write an OnBeforeExecute event handler to examine or alter an encoded method call before it is sent to the server. OnBeforeExecute occurs after the interface that the Converter (see page 4063) property specifies marshals a method call into a SOAP request packet, but before TRIO (see page 4056) calls the Execute method of the interface that the WebNode (see page 4065) property specifies.

You can use the SoapRequest parameter to read or alter the SOAP request before it is passed to the server. For example, if the Web Service provider publishes interoperability information, you can use this parameter to adjust SOAP packets to compensate for differences that the Web Service provider reports.

See Also

OnAfterExecute (see page 4065)

Execute

WebNode (see page 4065)

Converter (see page 4063)

OnSendAttachment (see page 4067)

70.1.1.7.3 TRIO.OnGetAttachment Event

Occurs when TRIO (see page 4056) reads each attachment included in the results from executing a method of the invokable interface.

Pascal

```
property OnGetAttachment: TOnGetAttachmentEvent;
```

C++

```
__property TOnGetAttachmentEvent OnGetAttachment;
```

Description

Write an OnGetAttachment event handler to examine the encoded attachments included in the results from executing a method call or to redirect them so that they are saved to a specific file. OnGetAttachment occurs once for each attachment received in the results from a call to the Execute method of the interface that the WebNode (see page 4065) property specifies. All OnGetAttachment events occur before the OnAfterExecute (see page 4065) event.

See Also

OnAfterExecute (see page 4065)

Execute

WebNode (see page 4065)

70.1.1.7.4 TRIO.OnSendAttachment Event

Occurs when TRIO (see page 4056) sends each attachment that is the value of an argument of the method call on the invokable interface.

Pascal

```
property OnSendAttachment: TOnSendAttachmentEvent;
```

C++

```
__property TOnSendAttachmentEvent OnSendAttachment;
```

Description

Write an OnSendAttachment event handler to examine or alter the attachments that are to be included in a request message that represents a method call. OnSendAttachment occurs once for each attachment sent with the request message to the Web Service provider. All OnSendAttachment events occur after the OnBeforeExecute (see page 4066) event.

See Also

OnBeforeExecute (see page 4066)

Execute

WebNode (see page 4065)

70.2 Interfaces

The following table lists interfaces in this documentation.

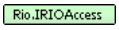
Interfaces

	Name	Description
	IRIOAccess (see page 4068)	IRIOAccess is the interface for accessing a remote interfaced object from its invokable interface.

70.2.1 IRIOAccess Interface

IRIOAccess is the interface for accessing a remote interfaced object from its invokable interface.

Class Hierarchy



Pascal

```
IRIOAccess = interface;
```

C++

```
__interface IRIOAccess;
```

File

Rio

Description

Remote interfaced objects (TRIO (see page 4056) and its descendants) implement IRIOAccess so that you can access the TRIO (see page 4056) instance given an invokable interface that it implements. This interface defines a single property, RIO (see page 4069), which returns the TRIO (see page 4056) instance that implements it.

Note: C++ method declarations that use IRIOAccess use the _di_IRIOAccess type instead. This type is a DelphiInterface wrapper around the IRIOAccess interface:

Note: typedef System::DelphiInterface< IRIOAccess > _di_IRIOAccess;

See Also

TRIO (see page 4056)

70.2.1.1 IRIOAccess Members

The following tables list the members exposed by IRIOAccess.

Public Methods

	Name	Description
	GetRIO (see page 4069)	Returns the value of the RIO (see page 4069) property.

Public Properties

	Name	Description
	RIO (see page 4069)	Provides access to the remote interfaced object that implements this interface.

70.2.1.2 IRIOAccess Methods

The methods of the IRIOAccess class are listed here.

Public Methods

	Name	Description
	GetRIO (see page 4069)	Returns the value of the RIO (see page 4069) property.

70.2.1.2.1 IRIOAccess.GetRIO Method

Returns the value of the RIO (see page 4069) property.

Pascal

```
function GetRIO: TRIO;
```

C++

```
__fastcall TRIO GetRIO();
```

Description

GetRIO is the read implementation for the RIO (see page 4069) property.

See Also

RIO (see page 4069)

70.2.1.3 IRIOAccess Properties

The properties of the IRIOAccess class are listed here.

Public Properties

	Name	Description
	RIO (see page 4069)	Provides access to the remote interfaced object that implements this interface.

70.2.1.3.1 IRIOAccess.RIO Property

Provides access to the remote interfaced object that implements this interface.

Pascal

```
property RIO: TRIO;
```

C++

```
__property TRIO RIO;
```

Description

Call GetRIO (see page 4069) to access the underlying remote interfaced object. This method is useful when the client of a Web Service uses the automatically-generated function to obtain an invokable interface rather than explicitly creating a TRIO (see page 4056) instance.

70.3 Types

The following table lists types in this documentation.

Types

Name	Description
TAfterExecuteEvent (see page 4070)	TAfterExecuteEvent is the type of event handlers that respond after a remote interfaced object receives the encoded results from executing a method call to a Web Service.
TBeforeExecuteEvent (see page 4070)	TBeforeExecuteEvent is the type of event handlers that respond before a remote interfaced object sends an encoded method call to a Web Service provider.

70.3.1 Rio.TAfterExecuteEvent Type

TAfterExecuteEvent is the type of event handlers that respond after a remote interfaced object receives the encoded results from executing a method call to a Web Service.

Pascal

```
TAfterExecuteEvent = procedure (const MethodName: string; SOAPResponse: TStream) of object;
```

C++

```
(const MethodName: string; SOAPResponse: TStream) ( TAfterExecuteEvent)();
```

File

Rio

Description

TAfterExecuteEvent is the type of the OnAfterExecute event handler, which is called after a client receives the encoded results from executing a method on an invokable interface, but before those results have been unmarshaled.

MethodName is the name of the method that was just executed on the server.

SOAPResponse is a stream from which you can read the encoded results that the server returned after executing the specified method.

See Also

Procedural types

Creating events

70.3.2 Rio.TBeforeExecuteEvent Type

TBeforeExecuteEvent is the type of event handlers that respond before a remote interfaced object sends an encoded method call to a Web Service provider.

Pascal

```
TBeforeExecuteEvent = procedure (const MethodName: string; SOAPRequest: TStream) of object;
```

C++

```
(const MethodName: string; SOAPRequest: TStream) ( TBeforeExecuteEvent)();
```

File

Rio

Description

TBeforeExecuteEvent is the type of the OnBeforeExecute event handler, which is called before a client sends a method call to a Web Service provider, but after it has marshaled the method call into a transportable string.

MethodName is the name of the method that is about to be executed on the server.

SOAPRequest is the text of the SOAP request packet. You can use this parameter to read or alter the SOAP request before it is passed to the server.

See Also

Procedural types

Creating events

70.4 Constants

The following table lists constants in this documentation.

Constants

Name	Description
StubSize (see page 4072)	This is constant Rio.StubSize.

70.4.1 Rio.StubSize Constant

Pascal

```
StubSize = 19;
```

C++

```
StubSize = 19;
```

File

Rio

Description

This is constant Rio.StubSize.

Index

_

_AddRef
- TEventDispatch class 2636
- TInvokableClass class 836
- TOleContainer class 2534
- TOleControl class 2646
- TOleServer class 2711
- TServerEventDispatch class 2721
- TXMLDataSet class 2167

_NewEnum
- ISharedPropertyGroupManager interface 2433

_Release
- TEventDispatch class 2636
- TInvokableClass class 837
- TOleContainer class 2535
- TOleControl class 2647
- TOleServer class 2712
- TServerEventDispatch class 2721
- TXMLDataSet class 2167

_SetColorProp
- TOleControl class 2647

_SetTColorProp
- TOleControl class 2648

_SetTFontProp
- TOleControl class 2648

_SetTOleEnumProp
- TOleControl class 2649

_SetTPictureProp
- TOleControl class 2649

A

aaDoubleClick enumeration member 2618

aaGetFocus enumeration member 2618

aaManual enumeration member 2618

Abort
- ITransactionContext interface 2434
- ITransactionContextEx interface 2436
- TPrinter class 3041

Aborted
- TPrinter class 3051

AbortOnKeyViol
- TIBBatchMove class 30

AbortOnProblem
- TIBBatchMove class 30

Access
- TRegistry class 3293

acInternal enumeration member 2521

acMultiInstance enumeration member 2521

acSingleInstance enumeration member 2521

Action
- TCustomQueryForm class 1846
- TIBControlAndQueryService class 420
- TIBLicensingService class 436
- TLimboTransactionInfo class 471
- TMenuItem class 1393
- TOptionItem class 3936
- TQueryForm class 1974
- TQuickAccessToolbarGroupItem class 3476

ActionAddUser enumeration member 480

ActionBar
- TQuickAccessToolbar class 3473

ActionChange
- TMenuItem class 1369

ActionDeleteUser enumeration member 480

ActionDisplayUser enumeration member 480

ActionLink
- TMenuItem class 1394

ActionManager
- TCustomRibbon class 3394
- TRibbon class 3484
- TRibbonApplicationMenuBar class 3950
- TRibbonGroup class 3503
- TRibbonGroupCollapsedControl class 3515
- TRibbonQuickAccessToolbar class 3548

ActionModifyUser enumeration member 480

Activate
IObjectControl interface 2425
TMtsAutoObject class 2389
TMtsDataModule class 2400
ActivateConnection
TIBCustomDataSet class 69
ActivateShadow
TIBConfigService class 414
ActivateTransaction
TIBCustomDataSet class 69
Active
TCustomListAction class 1024
TCustomRibbon class 3394
TIBCustomService class 430
TIBDataSet class 122
TIBQuery class 387
TIBTable class 615
TIBTransaction class 215
TStaticListAction class 1056
TVirtualListAction class 1068
ActiveChanged
TIBDataLink class 113
ActivePage
TCustomRibbon class 3394
ActiveTabFontColor
TCustomRibbonColorMap class 3402
TRibbonLunaColorMap class 3527
TRibbonObsidianColorMap class 3532
TRibbonSilverColorMap class 3551
ActiveUpdateException
TCustomProvider class 3114
TDataSetProvider class 3159
Actor
TSOAPHeader class 911
Add
TCustomImageList class 655
TCustomOutline class 2842
TCustomTabData class 3455
THeaderList class 825
TKeyTipList class 3594
TListControlItems class 1049
TMenuItem class 1369
TOptionItems class 3938
TPNGList class 2998
TPNGPointerList class 3001
TPopupList class 1428
TQuickAccessToolbarGroups class 3478
TRibbonTabs class 3561
TStringHash class 784
TWebRequestFiles class 3338
AddAttachment
IObjConverter interface 921
TSOAPDomConv class 2763
AddAttribute
TCustomPacketWriter class 3095
AddAttributes
TQueryRadioGroup class 1979
TQueryText class 2004
TQueryTextArea class 2014
TWebRadioGroupInput class 2125
TWebSelectOptionsInput class 2131
TWebTextAreaInput class 2142
TWebTextInput class 2148
AddChild
TCustomOutline class 2843
AddChildObject
TCustomOutline class 2844
AddColumn
TDataPacketWriter class 3138
AddConstraints
TDataPacketWriter class 3138
AddContainer
IVCLFrameForm interface 2616
AddDatabase
TIBTransaction class 209
AddDataSetAttributes
TDataPacketWriter class 3139
AddDynArrayToClear

TDataContext class 818
AddElements
TApplyUpdatesButton class 1800
TDataSetButton class 1862
TWebControlGroup class 2066
TWebForm class 2105
TWebListInput class 2117
AddEventNotifier
TIBDataBase class 180
AddExtraFieldProps
TDataPacketWriter class 3139
AddFieldLinks
TDataPacketWriter class 3139
AddGroup
TCustomRibbonPage class 3429
AddIcon
TCustomImageList class 655
AddImage
TCustomImageList class 656
AddImages
TCustomImageList class 656
AddIncludes
TBaseInetXPageProducer class 2261
AddIndex
TIBTable class 606
AddIndexDefs
TDataPacketWriter class 3139
AddItem
TListActionLink class 1039
AddLicense
TIBLicensingService class 435
AddMasked
TCustomImageList class 657
AddMultiRefNode
TSOAPDomConv class 2763
AddName
TIBXSQLDA class 540
AddNamespaceDecl
TSOAPDomConv class 2763
AddNode
TOutlineNode class 2944
AddObject
TCustomOutline class 2844
AddObjectAsWriting
TSOAPDomConv class 2763
AddObjectToDestroy
TDataContext class 819
AddOnDataChange
IInetXWebDataSet interface 2215
AddRecentItem
TCustomRibbon class 3370
AddRef
TAutoDispatch class 2502
TAutoObject class 2509
AddScriptComponents
TBaseInetXPageProducer class 2261
AddStrToClear
TDataContext class 819
AddTab
TCustomRibbon class 3370
AddTransaction
TIBDataBase class 180
AddUser
TIBSecurityService class 446
AddUStrToClear
TDataContext class 820
AddVariantToClear
TDataContext class 820
AddWStrToClear
TDataContext class 821
AdjustBiDiBehavior
TMenu class 1335
AdjustBounds
TRibbonGroupCollapsedControl class 3509
AdvancedDrawItem
TMenuItem class 1374
Advise
TLimboTransactionInfo class 471

AfterApplyUpdates
TCustomProvider class 3116
TDataSetProvider class 3164
AfterCancel
TIBCustomDataSet class 94
TIBDataSet class 122
AfterClose
TIBCustomDataSet class 94
TIBDataSet class 122
AfterConnect
TDispatchConnection class 1302
TIBDataBase class 198
TSharedConnection class 1315
AfterConstruction
TInvokableClass class 837
TRIO class 4060
AfterDatabaseDisconnect
TIBBase class 175
TIBCustomDataSet class 109
TIBDataSet class 133
TIBQuery class 394
TIBStoredProc class 595
TIBTable class 623
AfterDelete
TIBCustomDataSet class 94
TIBDataSet class 123
AfterDisconnect
TDispatchConnection class 1302
TIBDataBase class 199
TSharedConnection class 1315
AfterEdit
TIBCustomDataSet class 94
TIBDataSet class 123
AfterExecute
TCustomProvider class 3117
TDataSetProvider class 3165
AfterGetParams
TCustomProvider class 3118
TDataSetProvider class 3166
AfterGetRecords
TCustomProvider class 3118
TDataSetProvider class 3166
AfterInsert
TIBCustomDataSet class 95
TIBDataSet class 123
AfterOpen
TIBCustomDataSet class 95
TIBDataSet class 123
AfterPost
TIBCustomDataSet class 95
TIBDataSet class 123
AfterRefresh
TIBCustomDataSet class 95
AfterRowRequest
TCustomProvider class 3119
TDataSetProvider class 3167
AfterScroll
TIBCustomDataSet class 95
TIBDataSet class 124
AfterTransactionEnd
TIBBase class 175
TIBCustomDataSet class 109
TIBDataSet class 133
TIBQuery class 394
TIBStoredProc class 595
TIBTable class 623
AfterUpdateRecord
TBaseProvider class 3090
TDataSetProvider class 3168
Align
TCaptionAttributes class 1805
TCustomRibbonGalleryBar class 3984
TGridAttributes class 1926
TGridRowAttributes class 1932
TMaskEdit class 1104
TOleContainer class 2552
TOutline class 2878
TRibbon class 3484

TRibbonApplicationMenuBar class 3950
TRibbonSpinButton class 3801
AlignControls
TCustomRibbonGalleryBar class 3983
TCustomRibbonGalleryPopupBar class 3989
TCustomRibbonGroup class 3409
TRibbonApplicationPopupMenu class 3963
TRibbonStylePopupMenu class 3978
AlignGroupControls
IRibbonGroup interface 3568
TCustomRibbonGroup class 3410
AlignGroups
TCustomRibbonPage class 3429
Alignment
TMaskEdit class 1107
TPopupMenu class 1436
TRibbonComboBox class 3631
Allocation
TIBDatabaseInfo class 247
AllocBy
TCustomImageList class 675
AllocData
TDataContext class 821
AllocRecordBuffer
TIBCustomDataSet class 69
AllocServerData
TInvContext class 831
AllowActiveDoc
TOleContainer class 2556
AllowHotKeys
TRibbonBaseButtonControl class 3619
TRibbonRecentButtonControl class 3775
AllowInPlace
TOleContainer class 2556
AllowKeyboardNavigation
TCustomRibbon class 3395
AllowStreamedConnected
TIBDataBase class 189
Anchors
TMaskEdit class 1108
TMediaPlayer class 2332
TOleContainer class 2557
TOleControl class 2695
TRibbon class 3488
TRibbonSpinButton class 3804
TRibbonSpinEdit class 3845
ApplicationMenu
TCustomRibbon class 3395
TOptionItems class 3939
TRibbon class 3488
Apply
TIBDataSetUpdateObject class 136
TIBGeneratorField class 143
TIBUpdateSQL class 632
ApplyEvent
TIBGeneratorField class 145
ApplyUpdates
TCustomProvider class 3100
TCustomResolver class 3128
TIBCustomDataSet class 69
TIBDataBase class 180
AS_ApplyUpdates
TLocalAppServer class 3185
TMtsDataModule class 2401
AS_DataRequest
TLocalAppServer class 3185
TMtsDataModule class 2401
AS_Execute
TLocalAppServer class 3185
TMtsDataModule class 2402
AS_GetParams
TLocalAppServer class 3185
TMtsDataModule class 2403
AS_GetProviderNames
TLocalAppServer class 3186
TMtsDataModule class 2404
AS_GetRecords
TLocalAppServer class 3186

TMtsDataModule class 2404
AS_RowRequest
TLocalAppServer class 3186
TMtsDataModule class 2405
asbLeft enumeration member 3574
asbNone enumeration member 3574
asbRight enumeration member 3574
AsCurrency
TIBXSQLVAR class 548
AsDate
TIBXSQLVAR class 548
AsDateTime
TIBXSQLVAR class 549
AsDouble
TIBXSQLVAR class 549
AsFloat
TIBXSQLVAR class 549
AsInt64
TIBXSQLVAR class 550
AsInteger
TIBXSQLVAR class 550
AsLong
TIBXSQLVAR class 550
AsPointer
TIBXSQLVAR class 550
AsQuad
TIBXSQLVAR class 551
AsShort
TIBXSQLVAR class 551
Assign
TApplicationMenu class 3352
TCustomImageList class 657
TCustomRibbonGroup class 3410
TIBGeneratorField class 144
TIBXSQLVAR class 546
TJPEGImage class 998
TListControlItem class 1044
TOptionItem class 3935
TQuickAccessToolbar class 3472
TRibbonPopupGalleryBar class 3994
TRibbonTabItem class 3556
AssignClient
TMenuActionLink class 1354
AssignCurValues
TPacketDataSet class 3190
AssignStrings
TCustomQueryForm class 1843
AssignStringsCallback
TCustomQueryForm class 1844
AssignTo
TCustomImageList class 658
TGridAttributes class 1924
TGridRowAttributes class 1931
TJPEGImage class 999
TMenuItem class 1374
AsString
TIBXSQLVAR class 551
AsTime
TIBXSQLVAR class 551
AsTrimString
TIBXSQLVAR class 552
AsVariant
TIBXSQLVAR class 552
AsXSQLDA
TIBXSQLDA class 542
AsXSQLVAR
TIBXSQLVAR class 552
Attach
TIBCustomService class 427
Attachments
IOPConvert interface 2738
TOPToSoapDomConvert class 2756
AutoActivate
TOleContainer class 2557
AutoButtonSet
TMediaPlayer class 2312
AutoCalcFields
TIBCustomDataSet class 96

TIBDataSet class 124
AutoCheck
TMenuItem class 1394
AutoComplete
TMtsDataModule class 2411
AutoConnect
TOleServer class 2717
AutoDispatch
TAutoObject class 2510
AutoEnable
TMediaPlayer class 2333
AutoHotkeys
TMainMenu class 1327
TMenu class 1343
TMenuItem class 1394
TPopupMenu class 1436
AutoLineReduction
TMainMenu class 1328
TMenu class 1344
TMenuItem class 1395
TPopupMenu class 1437
AutoMerge
TMainMenu class 1328
AutoObject
TAutoDispatch class 2504
AutoObjectCount
TAutomation class 2506
AutoOpen
TMediaPlayer class 2336
AutoPopup
TPopupMenu class 1437
AutoRegister
TIBEvents class 263
AutoRegisterNativeTypes
TRemotableTypeRegistry class 891
AutoRewind
TMediaPlayer class 2337
AutoSelect
TMaskEdit class 1108
TRibbonComboBox class 3631
TRibbonSpinEdit class 3846
AutoSize
TMaskEdit class 1109
TRibbonSpinEdit class 3846
AutoSizingChanged
TCustomRibbonGroup class 3410
AutoStopAction
TIBTransaction class 215
AutoVerbMenu
TOleContainer class 2558
AvailableSpace
TCustomTabData class 3456

B

Back
TMediaPlayer class 2313
BackoutCount
TIBDatabaseInfo class 247
BackupFile
TIBBackupService class 411
TIBRestoreService class 442
BadIndex
TCustomOutline class 2845
BarHandleKeyTips
IRibbonBarKeyTip interface 3913
TCustomRibbonGroup class 3410
TCustomRibbonQuickAccessToolbar class 3444
BarHideKeyTips
IRibbonBarKeyTip interface 3913
TCustomRibbonGroup class 3411
TCustomRibbonQuickAccessToolbar class 3444
BarShowKeyTips
IRibbonBarKeyTip interface 3914
TCustomRibbonGroup class 3411
TCustomRibbonQuickAccessToolbar class 3445
BarWidth
TQuickAccessToolbar class 3474
BaseLevel

TIBDatabaseInfo class 248
BaseLocation
TConfigParams class 402
batAppend enumeration member 38
batAppendUpdate enumeration member 38
BatchInput
TIBDataSet class 119
TIBQuery class 382
TIBSQL class 521
BatchOutput
TIBDataSet class 119
TIBQuery class 382
TIBSQL class 522
batCopy enumeration member 38
batDelete enumeration member 38
batUpdate enumeration member 38
BeforeApplyUpdates
TCustomProvider class 3120
TDataSetProvider class 3168
BeforeCancel
TIBCustomDataSet class 96
TIBDataSet class 124
BeforeClose
TIBCustomDataSet class 96
TIBDataSet class 124
BeforeConnect
TDispatchConnection class 1303
TIBDataBase class 199
TSharedConnection class 1315
BeforeDatabaseDisconnect
TIBBase class 175
TIBCustomDataSet class 109
TIBDataSet class 133
TIBQuery class 394
TIBStoredProc class 596
TIBTable class 624
BeforeDelete
TIBCustomDataSet class 96
TIBDataSet class 125
BeforeDestruction
TInvokableClass class 838
TRIO class 4060
BeforeDisconnect
TDispatchConnection class 1303
TIBDataBase class 199
TSharedConnection class 1316
BeforeEdit
TIBCustomDataSet class 97
TIBDataSet class 125
BeforeExecute
TCustomProvider class 3121
TDataSetProvider class 3169
BeforeGetParams
TCustomProvider class 3121
TDataSetProvider class 3170
BeforeGetRecords
TCustomProvider class 3122
TDataSetProvider class 3170
BeforeInsert
TIBCustomDataSet class 97
TIBDataSet class 125
BeforeOpen
TIBCustomDataSet class 97
TIBDataSet class 125
BeforePost
TIBCustomDataSet class 97
TIBDataSet class 125
BeforeRefresh
TIBCustomDataSet class 97
BeforeRowRequest
TCustomProvider class 3123
TDataSetProvider class 3172
BeforeScroll
TIBCustomDataSet class 98
TIBDataSet class 126
BeforeTransactionEnd
TIBBase class 175
TIBCustomDataSet class 109

TIBDataSet class 134
TIBQuery class 395
TIBSQL class 522
TIBStoredProc class 596
TIBTable class 624
BeforeUpdateRecord
TBaseProvider class 3090
TDataSetProvider class 3172
BeginAutoDrag
TCustomRibbonComboBox class 3585
BeginDoc
TPrinter class 3043
BeginUpdate
TCustomOutline class 2846
TCustomResolver class 3128
TCustomRibbon class 3370
TDataSetResolver class 3178
BestTabWidth
TRibbonTabItem class 3558
BevelEdges
TMaskEdit class 1110
BevelInner
TMaskEdit class 1110
BevelKind
TMaskEdit class 1111
BevelMinimizedInner
TCustomRibbonColorMap class 3403
TRibbonLunaColorMap class 3527
TRibbonObsidianColorMap class 3533
TRibbonSilverColorMap class 3551
BevelMinimizedOuter
TCustomRibbonColorMap class 3403
TRibbonLunaColorMap class 3527
TRibbonObsidianColorMap class 3533
TRibbonSilverColorMap class 3551
BevelOuter
TMaskEdit class 1111
BevelWidth
TMaskEdit class 1112
BgColor
TGridAttributes class 1926
TGridRowAttributes class 1932
BiDiMode
TMainMenu class 1329
TMaskEdit class 1112
TMenu class 1344
TPopupMenu class 1438
TRibbon class 3488
TRibbonGroup class 3503
TRibbonPage class 3538
Bitmap
TCustomRibbonSkin class 4022
TJPEGImage class 1008
TMenuItem class 1395
BkColor
TCustomImageList class 676
BlendColor
TCustomImageList class 676
BlobID
TIBBlobStream class 50
BlobMaxSegmentSize
TIBBlobStream class 51
BlobNumSegments
TIBBlobStream class 51
BlobSize
TIBBlobStream class 51
BlobType
TIBBlobStream class 51
BlockingFactor
TIBBackupService class 411
BOAInit
IORB interface 2806
BodyContent
TInetXPageElements class 2285
Bof
TIBSQL class 530
BookmarkValid
TIBCustomDataSet class 70

Border
TGridAttributes class 1926
BorderStyle
TMaskEdit class 1112
TOleContainer class 2558
TOutline class 2881
Break
TMenuItem class 1396
BreakButtons
TFormLayout class 1922
BringDatabaseOnline
TIBConfigService class 414
BrokerName
TOLEnterpriseConnection class 1308
BrowseProperties
TOleControl class 2649
btBack enumeration member 2376
btEject enumeration member 2376
btNext enumeration member 2376
BtnSelectedFont
TRibbonLunaColorMap class 3527
TRibbonObsidianColorMap class 3533
TRibbonSilverColorMap class 3551
btPause enumeration member 2376
btPlay enumeration member 2376
btPrev enumeration member 2376
btRecord enumeration member 2376
btStep enumeration member 2376
btStop enumeration member 2376
Buffer
IMarshalInBuffer interface 2797
IMarshalOutBuffer interface 2802
BufferChunks
TIBCustomDataSet class 98
TIBDataSet class 126
TIBQuery class 387
TIBTable class 615
BufferContent
TMultipartContentParser class 3344
BufferSize
TIBControlAndQueryService class 420
TIBCustomService class 430
Button
TRibbonSpinEdit class 3847
ButtonStyle
TTimerSpeedButton class 3910
ByChunk enumeration member 478
ByLine enumeration member 478
ByName
TIBXSQLDA class 540

C

CachedUpdates
TIBCustomDataSet class 98
TIBDataSet class 126
TIBQuery class 388
TIBTable class 615
CachedUpdateStatus
TIBCustomDataSet class 70
CacheFile
TSOAPAttachment class 904
CacheFilePersist
TSOAPAttachment class 905
CalcBounds
TRibbonApplicationButton class 3603
TRibbonApplicationMenuButton class 3606
TRibbonApplicationMenuItem class 3614
TRibbonBaseButtonControl class 3619
TRibbonCheckBoxControl class 3627
TRibbonComboBoxItem class 3683
TRibbonComboControl class 3687
TRibbonCustomControl class 3700
TRibbonCustomizeItem class 3711
TRibbonCustomStatusControl class 3714
TRibbonDropDownButton class 3717
TRibbonGalleryControlMenuItem class 3723
TRibbonGalleryItem class 3730
TRibbonGalleryMenuItem class 3731

TRibbonGripItem class 3737
TRibbonLargeButtonControl class 3744
TRibbonLargeSplitButton class 3747
TRibbonMainMenuSeparator class 3755
TRibbonMenuItem class 3757
TRibbonMenuSeparatorItem class 3761
TRibbonQATScrollButton class 3768
TRibbonRadioButtonControl class 3773
TRibbonRichContentItem class 3777
TRibbonRichContentMainMenuItem class 3782
TRibbonSeparator class 3785
TRibbonSmallButtonControl class 3788
TRibbonSmallSplitButton class 3792
TRibbonSplitMenuItem class 3899
TRibbonTextControl class 3904
CalcLayout
TRibbonDropDownButton class 3717
TRibbonGalleryItem class 3730
TRibbonLargeButtonControl class 3744
TRibbonLargeSplitButton class 3748
TRibbonMenuItem class 3757
TRibbonRichContentMainMenuItem class 3782
TRibbonSmallSplitButton class 3792
CalculateTabWidths
TCustomTabData class 3455
Call
TIBBlobStream class 43
TIBDataBase class 181
TIBDatabaseInfo class 238
TIBSetup class 318
TIBSQL class 522
TIBTransaction class 209
CallComplete
TMtsDataModule class 2406
CanAddClass
TBaseInetXPageProducer class 2261
TWebControlGroup class 2067
TWebForm class 2105
CanAddToQuickAccessToolbar
IRibbonControl interface 3916
TRibbonBaseButtonControl class 3620
TRibbonComboControl class 3687
TRibbonCustomControl class 3701
TRibbonGalleryControl class 3721
TRibbonGalleryControlMenuItem class 3724
CanAutoSize
TRibbonStylePopupMenu class 3979
CanBePooled
IObjectControl interface 2425
TMtsAutoObject class 2390
TMtsDataModule class 2406
CanCalculateHeight
TRibbonRichContentItem class 3778
Cancel
TIBBlobStream class 43
CancelUpdates
TIBCustomDataSet class 70
CanCollapse
TCustomRibbonGroup class 3421
CanInPlaceActivate
TOleControl class 2650
CanParse
TMultipartContentParser class 3344
CanPaste
TOleContainer class 2558
Canvas
TCustomOutline class 2862
TPrinter class 3053
Capabilities
TMediaPlayer class 2337
TPrinter class 3053
CapabilityMask
TLicenseMaskInfo class 469
capAbove enumeration member 2237
capBelow enumeration member 2237
capLeft enumeration member 2237
capRight enumeration member 2237
Caption

TApplicationMenu class 3353
TApplyUpdatesButton class 1801
TDataSetButton class 1864
TFieldRadioGroup class 1878
TFieldSelectOptions class 1888
TFieldStatus class 1896
TFieldText class 1901
TFieldTextArea class 1908
TListControlItem class 1046
TMenuItem class 1396
TOleContainer class 2559
TOptionItem class 3937
TQueryButton class 1959
TQueryRadioGroup class 1981
TQuerySelectOptions class 1993
TQueryText class 2005
TQueryTextArea class 2016
TQuickAccessToolbarGroupItem class 3477
TRibbon class 3489
TRibbonGroup class 3503
TRibbonGroupCollapsedControl class 3515
TRibbonPage class 3539
TRibbonRecentDocuments class 3972
TRibbonTabItem class 3558
TSelectOptionsColumn class 2028
TStaticListAction class 1057
TStatusColumn class 2036
TTextAreaColumn class 2044
TTextColumn class 2052
TVirtualListAction class 1068
TWebButton class 2062
TWebDataDisplay class 2083
TWebStatus class 2137
CaptionAttributes
TFieldRadioGroup class 1878
TFieldSelectOptions class 1888
TFieldStatus class 1896
TFieldText class 1902
TFieldTextArea class 1908
TQueryRadioGroup class 1981
TQuerySelectOptions class 1994
TQueryText class 2005
TQueryTextArea class 2016
TSelectOptionsColumn class 2028
TStatusColumn class 2036
TTextAreaColumn class 2044
TTextColumn class 2052
TWebDataDisplay class 2084
TWebStatus class 2137
CaptionFontColor
TCustomRibbonColorMap class 3403
TRibbonLunaColorMap class 3528
TRibbonObsidianColorMap class 3533
TRibbonSilverColorMap class 3552
CaptionPosition
TFieldRadioGroup class 1878
TFieldSelectOptions class 1888
TFieldStatus class 1897
TFieldText class 1902
TFieldTextArea class 1909
TQueryRadioGroup class 1982
TQuerySelectOptions class 1994
TQueryText class 2006
TQueryTextArea class 2017
TWebDataDisplay class 2084
TWebStatus class 2138
CaseSensitive
TListControlItems class 1052
TMemIniFile class 782
ccCdecl enumeration member 797, 2456
ccPascal enumeration member 797, 2456
ccReg enumeration member 797
ccRegister enumeration member 2456
ccSafeCall enumeration member 797, 2456
ccStdCall enumeration member 797, 2456
cd16Bit enumeration member 685
cd24Bit enumeration member 685
cd32Bit enumeration member 685

cd4Bit enumeration member 685
cd8Bit enumeration member 685
cdDefault enumeration member 685
cdDeviceDependent enumeration member 685
CellPadding
TGridAttributes class 1927
CellSpacing
TGridAttributes class 1927
Change
TChangeLink class 649
TCustomImageList class 658
Changed
THashedStringList class 751
TJPEGImage class 1000
TListControlItem class 1044
TOleContainer class 2535
TWebDisplay class 2096
TXMLData class 2159
ChangedCount
TIBBatchMove class 31
ChangedDataSetField
TXMLDataSet class 2167
TXMLDataSetParent class 2174
ChangedDisplayComponent
TWebDisplay class 2096
TXMLDisplayParent class 2190
ChangedTableName
TIBBatchMove class 31
ChangedUseParent
TXMLDataParent class 2163
TXMLDataSetParent class 2174
TXMLDisplayParent class 2190
ChangedXMLBroker
TXMLData class 2159
TXMLDataParent class 2163
TXMLDataSet class 2167
TXMLDataSetParent class 2174
ChangeFieldControls
TWebControlGroup class 2067
ChangeIconDialog
TOleContainer class 2535
ChangeKey
TRegistry class 3255
ChangeLevelBy
TOutlineNode class 2944
CharCase
TMaskEdit class 1113
TRibbonComboBox class 3632
Check
TCustomPacketWriter class 3095
CheckActive
TIBCustomService class 427
TIBDataBase class 181
CheckAutoStop
TIBTransaction class 210
CheckBrowseMode
TIBDataLink class 113
CheckClosed
TIBSQL class 522
CheckCursor
TCustomMaskEdit class 1083
CheckDatabase
TIBBase class 170
CheckDatabaseName
TIBDataBase class 181
CheckDatabasesInList
TIBTransaction class 210
CheckDataSet
TDataSetProvider class 3149
CheckDatasetClosed
TIBCustomDataSet class 71
CheckDatasetOpen
TIBCustomDataSet class 71
CheckDB enumeration member 485
Checked
TMenuItem class 1401
CheckEncodingStyle
TSOAPDomConv class 2764

CheckInactive
TIBCustomService class 428
TIBDataBase class 181
CheckInTransaction
TIBTransaction class 210
CheckNotInTransaction
TIBTransaction class 210
CheckNotUniDirectional
TIBCustomDataSet class 71
CheckOpen
TIBSQL class 523
CheckReadable
TIBBlobStream class 43
CheckResolver
TBaseProvider class 3081
CheckRibbonSize
TCustomRibbon class 3370
CheckTransaction
TIBBase class 170
CheckTypeSize
TIBBCDField class 60
TIBStringField class 147
CheckValidStatement
TIBSQL class 523
CheckWritable
TIBBlobStream class 44
ChildName
TSharedConnection class 1314
ChildNodesAreNull
TSOAPDomConv class 2764
ckAttachToInterface enumeration member 2725
ckNewInstance enumeration member 2725
ckRemote enumeration member 2725
ckRunningInstance enumeration member 2725
ckRunningOrNew enumeration member 2725
ClassOptions
TRemotableTypeRegistry class 877
ClassToURI
TRemotableTypeRegistry class 878
Clean
TQuickAccessToolbarGroups class 3479
Clear
TCustomImageList class 658
TCustomMaskEdit class 1083
TCustomOutline class 2846
TCustomTabData class 3456
THeaderList class 826
TIBXSQLVAR class 546
TKeyTipList class 3594
TMemIniFile class 771
TMenuItem class 1375
TStringHash class 784
TUpdateTree class 3207
ClearBorderSpace
IVCLFrameForm interface 2616
ClearCalcFields
TIBCustomDataSet class 71
ClearItem
TMenuItemStack class 1426
ClearRecentItems
TCustomRibbon class 3371
Click
TCustomOutline class 2846
TMediaPlayer class 2313
TMenuItem class 1375
TRibbonBaseButtonControl class 3620
TRibbonComboControl class 3688
TRibbonCustomControl class 3701
TRibbonDropDownButton class 3717
TRibbonGroupCollapsedControl class 3509
TRibbonLargeSplitButton class 3748
TRibbonMenuItem class 3757
TRibbonQATScrollButton class 3768
TRibbonSmallSplitButton class 3793
TRibbonSplitMenuItem class 3899
Clients
TCustomStaticListAction class 1032
Clone

IEnumNames interface 2418
TCustomRibbonGroup class 3411
Close
TIBSQL class 523
TMediaPlayer class 2314
TOleContainer class 2536
CloseBlob
TIBBlobStream class 44
CloseDataSets
TIBDataBase class 182
CloseKey
TRegistry class 3255
CloseMenu
TRibbonApplicationMenuBar class 3945
CloseUp
IRibbonGroupDropdown interface 3917
TRibbonGroupCollapsedControl class 3509
TRibbonGroupDropDown class 3740
CMBidimodechanged
TCustomRibbon class 3371
TRibbonComboControl class 3688
CMControlChange
TCustomActionControlBar class 3357
CMControlListChanging
TCustomRibbonPage class 3429
cmDBMgmt enumeration member 328
cmDBQuery enumeration member 328
CMDesignHitTest
TCustomRibbon class 3371
TRibbonGroupScrollButton class 3523
CmdLineTools
TInstallOptions class 325
CMFontchanged
TCustomRibbon class 3371
TCustomRibbonGroup class 3412
TCustomRibbonQuickAccessToolbar class 3445
TRibbonGroupCollapsedControl class 3510
CMItemDropPoint
TCustomRibbonGroup class 3412
CMMouseenter
TCustomRibbon class 3371
TCustomRibbonComboBox class 3585
TCustomRibbonGroup class 3412
TOffice2007Button class 3463
TRibbonComboControl class 3688
TRibbonGroupCollapsedControl class 3510
TRibbonGroupScrollButton class 3523
TRibbonSpinEdit class 3839
TTimerSpeedButton class 3908
CMMouseEnter
TRibbonApplicationMenuButton class 3607
TRibbonQATScrollButton class 3769
CMMouseleave
TCustomRibbon class 3372
TCustomRibbonComboBox class 3586
TCustomRibbonGroup class 3412
TOffice2007Button class 3463
TRibbonComboControl class 3688
TRibbonGroupCollapsedControl class 3510
TRibbonGroupScrollButton class 3523
TRibbonLargeSplitButton class 3748
TRibbonSpinEdit class 3840
TTimerSpeedButton class 3908
CMMouseLeave
TRibbonApplicationMenuButton class 3607
TRibbonQATScrollButton class 3769
TRibbonSmallSplitButton class 3793
CMTextchanged
TCustomRibbon class 3372
TCustomRibbonGroup class 3413
TCustomRibbonPage class 3429
TRibbonApplicationPopupMenu class 3963
TRibbonGroupCollapsedControl class 3510
CMThemeChanged
TCustomRibbon class 3372
cmUsrMgmt enumeration member 328
CMVisiblechanged
TRibbonComboControl class 3689

TRibbonCustomControl class 3701
cnJDBC enumeration member 329
cnODBC enumeration member 329
cnOLEDB enumeration member 329
ColDelimiter
TIBInputDelimitedFile class 504
TIBOutputDelimitedFile class 510
Collapse
TCustomOutline class 2847
TOutlineNode class 2944
CollapsedControl
TCustomRibbonGroup class 3421
CollapseGroup
TCustomRibbonPage class 3430
CollapseGroupCount
TCustomRibbonPage class 3437
Color
TCustomRibbonComboBox class 3591
TMaskEdit class 1113
TOleContainer class 2561
TOutline class 2881
TRibbonSpinEdit class 3847
ColorDepth
TCustomImageList class 677
ColoredButtons
TMediaPlayer class 2337
ColorMap
TCustomRibbon class 3395
ColumnCount
TFormLayout class 1922
Columns
TIBBatch class 496
ComboBox
TRibbonComboControl class 3697
ComboClick
TRibbonComboControl class 3689
Command
TMenuItem class 1401
CommandType
TApplicationMenu class 3354
Commit
ITransactionContext interface 2434
ITransactionContextEx interface 2436
TIBTransaction class 211
CommitAction enumeration member 482
CommitAdvise enumeration member 483
CommitCount
TIBBatchMove class 31
CommitGlobal enumeration member 483
CommitRetaining
TIBTransaction class 211
CommitState enumeration member 484
CompareBookmarks
TIBCustomDataSet class 71
CompareItems
TListControlItems class 1050
CompareXMLData
TXMLDataSet class 2167
Compress
TJPEGImage class 1000
CompressionQuality
TJPEGImage class 1008
ComputerName
TDCOMConnection class 1287
TOLEnterpriseConnection class 1308
TServerItem class 2472
ConfigFileData
TConfigParams class 403
ConfigFileKey
TConfigFileData class 401
ConfigFileParams
TConfigParams class 403
ConfigFileValue
TConfigFileData class 401
ConfigParameters enumeration member 479
ConfigParams
TIBServerProperties class 456
Connect

TOleServer class 2712
Connected
TDispatchConnection class 1296
TIBDataBase class 189
ConnectEvent
TSharedConnection class 1311
ConnectEvents
TOleServer class 2713
ConnectionChange
TXMLDataSet class 2168
ConnectivityClients
TInstallOptions class 326
ConnectKind
TOleServer class 2718
ConnectType
TMidasConnection class 1786
Constraints
TDataPacketWriter class 3144
TDataSetProvider class 3159
TIBQuery class 388
TIBTable class 616
TMaskEdit class 1114
TMediaPlayer class 2338
TOleContainer class 2562
TRibbonSpinButton class 3805
TRibbonSpinEdit class 3848
ConstraintsStored
TIBCustomDataSet class 72
ContainedControl
IRibbonContainedControl interface 3915
TRibbonComboControl class 3697
TRibbonCustomControl class 3710
TRibbonGalleryControl class 3722
TRibbonGalleryControlMenuItem class 3728
Content
TCustomInetXPageProducer class 2279
TWebButton class 2060
TWebControlGroup class 2068
TWebDataDisplay class 2076
TWebForm class 2106
ContentFromStream
TBaseInetXPageProducer class 2262
ContentType
ITypeCode interface 2817
TSOAPAttachment class 905
ContextMenuPopupHandler
TCustomRibbon class 3372
ContextSensitiveHelp
TOleControl class 2650
Control
TEventDispatch class 2638
TKeyTipMatchItem class 3599
TRibbonGroupDockForm class 3521
TRibbonSpinButton class 3798
ControlColor
TCustomRibbonColorMap class 3403
TRibbonLunaColorMap class 3528
TRibbonObsidianColorMap class 3533
TRibbonSilverColorMap class 3552
ControlContent
TQueryTextArea class 2014
TWebDataDisplay class 2077
TWebListInput class 2117
TWebStatus class 2135
TWebTextAreaInput class 2142
TWebTextInput class 2148
ControlData
TOleControl class 2696
ControlFrameColor
TCustomRibbonColorMap class 3404
TRibbonLunaColorMap class 3528
TRibbonObsidianColorMap class 3534
TRibbonSilverColorMap class 3552
ControlInactiveColor
TCustomRibbonColorMap class 3404
TRibbonLunaColorMap class 3528
TRibbonObsidianColorMap class 3534
TRibbonSilverColorMap class 3552

ConvertByteArrayToSoap
TSOAPDomConv class 2764
ConvertEnumToSoap
TSOAPDomConv class 2764
Converter
TRIO class 4063
ConvertExtTables enumeration member 477
ConvertNativeArrayToSoap
TSOAPDomConv class 2765
ConvertNativeDataToSoap
TSOAPDomConv class 2765
ConvertObjectToSOAP
TSOAPDomConv class 2765
ConvertSoapToEnum
TSOAPDomConv class 2766
ConvertSoapToNativeArray
TSOAPDomConv class 2766
ConvertSoapToNativeArrayElem
TSOAPDomConv class 2766
ConvertSoapToNativeData
TSOAPDomConv class 2766
ConvertSOAPToObject
TSOAPDomConv class 2767
ConvertSoapToVariant
TSOAPDomConv class 2767
ConvertVariantToSoap
TSOAPDomConv class 2768
Copies
TPrinter class 3054
Copy
TOleContainer class 2536
CopyFrom
TStream class 3006
CopyOnSave
TOleContainer class 2563
CopyParams
TIBStoredProc class 588
CorbaObject
ICorbaObj interface 2796
Count
IGetContextProperties interface 2419
TCustomImageList class 677
TCustomListAction class 1024
TCustomStaticListAction class 1032
TCustomVirtualListAction class 1036
THeaderList class 828
TIBXSQLDA class 543
TKeyTipList class 3594
TMenuItem class 1402
TPNGPointerList class 3002
Create
EDSWriter class 3076
EIBError class 4
EIBInstall class 308
EOleException class 2498
EOleSysError class 2500
ERemotableException class 811
Exception class 2993
TAlphaBitmap class 4018
TApplicationMenu class 3351
TApplyUpdatesButton class 1799
TAutoDispatch class 2501
TAutomation class 2505
TAutoObject class 2508
TBaseInetXPageProducer class 2259
TBasePageItemsProducer class 2967
TBaseProvider class 3079
TCaptionAttributes class 1805
TConfigFileData class 400
TConfigParams class 402
TCustomActionControlBar class 3356
TCustomDataGrid class 1814
TCustomFieldGroup class 1827
TCustomImageList class 653
TCustomInetXPageProducer class 2278
TCustomIniFile class 720
TCustomLayoutGroup class 1832
TCustomListAction class 1020

TCustomMaskEdit class 1082
TCustomOutline class 2840
TCustomPacketWriter class 3094
TCustomProvider class 3099
TCustomQueryForm class 1843
TCustomResolver class 3126
TCustomRibbon class 3362
TCustomRibbonComboBox class 3584
TCustomRibbonGalleryBar class 3983
TCustomRibbonGroup class 3407
TCustomRibbonPage class 3426
TCustomRibbonQuickAccessPopupToolbar class 3439
TCustomRibbonQuickAccessToolbar class 3443
TCustomRibbonSkin class 4019
TCustomStaticListAction class 1029
TCustomTabData class 3454
TDatabaseInfo class 405
TDataContext class 815
TDataSetProvider class 3148
TDataSetResolver class 3178
TDCOMConnection class 1286
TDeleteButton class 1869
TDispatchAppServer class 1289
TDispatchConnection class 1291
TEnumPropDesc class 2632
TEventDispatch class 2635
TFileStream class 2994
TFirstButton class 1915
TFormLayout class 1916
TGridAttributes class 1924
TGridRowAttributes class 1930
THeaderList class 824
TIBBackupService class 410
TIBBase class 166
TIBBatchMove class 27
TIBBCDField class 60
TIBBlobStream class 41
TIBControlAndQueryService class 419
TIBControlService class 422
TIBCustomDataSet class 66
TIBCustomService class 426
TIBCustomSQLMonitor class 562
TIBDataBase class 178
TIBDatabaseInfo class 234
TIBDataLink class 112
TIBDataSetUpdateObject class 135
TIBDSBlobStream class 139
TIBEvents class 259
TIBExtract class 272
TIBGeneratorField class 143
TIBInstall class 311
TIBQuery class 381
TIBRestoreService class 440
TIBSecurityService class 445
TIBServerProperties class 453
TIBSetup class 318
TIBSQL class 516
TIBStoredProc class 586
TIBStringField class 147
TIBTable class 605
TIBTimer class 203
TIBTransaction class 208
TIBUpdateSQL class 631
TIBValidationService class 462
TIBXSQLDA class 537
TIBXSQLVAR class 545
TInsertButton class 1937
TInvokableClass class 835
TInvokableClassRegistry class 842
TISAPIApplication class 961
TISAPIFactory class 963
TISAPIRequest class 971
TISAPIResponse class 978
TJPEGImage class 995
TKeyTipList class 3593
TKeyTipListEnumerator class 3597
TKeyTipMatchItem class 3599
TLastButton class 1939

TLicenseInfo class 467
TListControlItem class 1042
TListControlItems class 1049
TLocalAppServer class 3183
TMask class 1179
TMediaPlayer class 2311
TMemIniFile class 770
TMenu class 1333
TMenuItem class 1366
TMenuItemEnumerator class 1424
TMidasConnection class 1783
TMtsDataModule class 2399
TNextButton class 1944
TNextPageButton class 1947
TObjectDispatch class 2485
TOffice2007Button class 3462
TOleContainer class 2532
TOleControl class 2643
TOleForm class 2614
TOleServer class 2709
TOPToSoapDomConvert class 2751
TOutlineNode class 2942
TPacketDataSet class 3189
TPageItemsProducer class 2971
TPanningWindow class 694
TPNGPointerList class 3000
TPopupMenu class 1432
TPostButton class 1949
TPrinter class 3040
TPriorButton class 1951
TPriorPageButton class 1953
TQueryButton class 1955
TQueryText class 2003
TQueryTextArea class 2013
TQuickAccessToolbar class 3470
TQuickAccessToolbarGroupItem class 3476
TQuickAccessToolbarGroups class 3478
TRegIniFile class 3239
TRegistry class 3251
TRegistryClass class 2512
TRegistryIniFile class 3304
TRemotable class 870
TRemotableTypeRegistry class 876
TResetQueryButton class 2023
TResourceStream class 3003
TRibbonApplicationMenuBar class 3944
TRibbonApplicationMenuButton class 3605
TRibbonApplicationMenuItem class 3613
TRibbonApplicationPopupMenu class 3962
TRibbonBaseButtonControl class 3617
TRibbonComboControl class 3685
TRibbonCustomControl class 3699
TRibbonGripItem class 3736
TRibbonGroupCollapsedControl class 3507
TRibbonGroupDockForm class 3519
TRibbonGroupScrollButton class 3522
TRibbonMenuItem class 3756
TRibbonMenuSmallButtonControl class 3763
TRibbonRecentDocuments class 3971
TRibbonRichContentItem class 3777
TRibbonRichContentMainMenuItem class 3781
TRibbonSeparator class 3784
TRibbonSmallSplitButton class 3790
TRibbonSpinButton class 3797
TRibbonSpinEdit class 3838
TRibbonSplitMenuItem class 3898
TRibbonStyleActionBars class 4023
TRibbonStyleMainPopupMenu class 3976
TRibbonStylePopupMenu class 3977
TRibbonTabs class 3561
TRIO class 4058
TServerCollection class 2467
TServerEventDispatch class 2720
TServerItem class 2471
TSharedConnection class 1310
TSimpleObjectBroker class 2476
TSOAPAttachment class 897
TSOAPDomConv class 2760

TSQLResolver class 3198
TStringHash class 783
TSubmitQueryButton class 2040
TTimerSpeedButton class 3907
TUndoButton class 2057
TUpdateTree class 3206
TWebButton class 2059
TWebControlGroup class 2065
TWebDataDisplay class 2075
TWebDisplay class 2095
TWebDisplayReferenceGroup class 2100
TWebForm class 2104
TWebListInput class 2116
TWebRadioGroupInput class 2125
TWebRequestFile class 3333
TWebRequestFiles class 3336
TWebRequestFileStream class 3340
TWebSelectOptionsInput class 2131
TWebStatus class 2135
TWebTextAreaInput class 2141
TWebTextInput class 2147
TXMLButton class 2153
TXMLData class 2158
TXMLDataSet class 2165
TXMLDisplayGroup class 2178
TXMLDisplayParent class 2189
TXMLDisplayReferenceButton class 2193
CreateAliasTC
IORB interface 2806
CreateArrayTC
IORB interface 2806
CreateAutoDispatch
TAutoObject class 2510
CreateBlob
TIBBlobStream class 44
CreateBlobStream
TIBCustomDataSet class 72
CreateContextMenu
TCustomRibbon class 3373
CreateControl
IRibbonGroup interface 3569
TCustomActionControlBar class 3357
TCustomRibbonGalleryBar class 3984
TCustomRibbonGroup class 3413
TCustomRibbonQuickAccessPopupToolbar class 3440
TCustomRibbonQuickAccessToolbar class 3445
TRibbonActionPopupMenu class 3941
TRibbonApplicationMenuBar class 3945
TRibbonStylePopupMenu class 3979
CreateControls
TCustomActionControlBar class 3357
TCustomRibbonGalleryPopupBar class 3990
TCustomRibbonGroup class 3413
TCustomRibbonQuickAccessToolbar class 3446
TRibbonApplicationMenuBar class 3946
TRibbonApplicationPopupMenu class 3963
CreateDatabase
TIBDataBase class 182
CreateDataPacket
TBaseProvider class 3081
TDataSetProvider class 3149
CreateDataSet
TXMLDataSet class 2168
CreateDefaultFields
TXMLDisplayGroup class 2179
CreateFields
TIBCustomDataSet class 72
CreateFromDelta
TPacketDataSet class 3191
CreateInstance
IObjectContext interface 2421
ITransactionContext interface 2435
ITransactionContextEx interface 2436
CreateKey
TRegistry class 3258
CreateLinkToFile
TOleContainer class 2537
CreateMenuItem

TMenu class 1335
CreateMultiRefNode
TSOAPDomConv class 2768
CreateNewDB enumeration member 480
CreateNULLNode
TSOAPDomConv class 2768
CreateObject
TOleContainer class 2537
CreateObjectFromFile
TOleContainer class 2538
CreateObjectFromInfo
TOleContainer class 2539
CreateObjectNode
TSOAPDomConv class 2768
CreateObjRefTC
IORB interface 2807
CreateOptionsItems
TRibbonApplicationPopupMenu class 3964
CreatePageElements
TBaseInetXPageProducer class 2262
CreateParams
TOleContainer class 2540
TPanningWindow class 695
TRibbonGroupDockForm class 3520
TRibbonPopupGalleryBar class 3994
TRibbonSpinEdit class 3840
CreateProperty
ISharedPropertyGroup interface 2430
CreatePropertyByPosition
ISharedPropertyGroup interface 2430
CreatePropertyGroup
ISharedPropertyGroupManager interface 2432
CreateRecentDocuments
TRibbonApplicationPopupMenu class 3964
CreateRequest
IStub interface 2813
CreateResolver
TBaseProvider class 3082
TDataSetProvider class 3150
CreateScalarNodeXS
TSOAPDomConv class 2769
CreateScriptManager
TBaseInetXPageProducer class 2262
CreateScrollButton
TCustomRibbonPage class 3430
CreateSequenceTC
IORB interface 2807
CreateSize
TCustomImageList class 653
CreateStructTC
IORB interface 2807
CreateTable
TIBTable class 606
CreateTC
IORB interface 2807
CreateTypedNode
TSOAPDomConv class 2769
CreateWnd
TCustomRibbonQuickAccessPopupToolbar class 3440
TOleControl class 2651
TRibbonApplicationPopupMenu class 3964
TRibbonSpinEdit class 3840
CreatingControls
TCustomActionControlBar class 3358
cRibbonFirstTabOffSet
TCustomRibbon class 3364
cRibbonGroupCaptionHeight
TCustomRibbon class 3364
cRibbonGroupHeight
TCustomRibbon class 3364
cRibbonGroupLeftOffset
TCustomRibbonPage class 3428
cRibbonGroupSpacing
TCustomRibbonPage class 3428
cRibbonHeight
TCustomRibbon class 3364
cRibbonHideHeight
TCustomRibbon class 3365

cRibbonHideWidth
TCustomRibbon class 3365
cRibbonMinimizedHeight
TCustomRibbon class 3365
cRibbonMinimumCaptionWidth
TCustomRibbon class 3365
cRibbonPageHeight
TCustomRibbon class 3366
cRibbonQATHeight
TCustomRibbon class 3366
cRibbonQuickAccessToolbarLeft
TCustomRibbon class 3366
cRibbonQuickAccessToolbarTop
TCustomRibbon class 3366
cRibbonTabHeight
TCustomRibbon class 3366
cRibbonTabScrollButtonWidth
TCustomRibbon class 3367
cRibbonTabSpacing
TCustomRibbon class 3367
cRibbonUnthemedCaptionHeight
TCustomRibbon class 3367
ctCommands enumeration member 3571
ctDCOM enumeration member 1789
ctFromData enumeration member 2619
ctFromFile enumeration member 2619
Ctl3D
TMaskEdit class 1115
TOleContainer class 2563
TOutline class 2882
TRibbonSpinButton class 3806
TRibbonSpinEdit class 3849
ctLinkFromData enumeration member 2619
ctLinkToFile enumeration member 2619
ctNewObject enumeration member 2619
ctOLEnterprise enumeration member 1789
ctRecent enumeration member 3571
ctSockets enumeration member 1789
Current
TIBCustomDataSet class 73
TIBSQL class 523
TKeyTipList class 3595
TKeyTipListEnumerator class 3598
TMenuItemEnumerator class 1425
CurrentDBKey
TIBTable class 616
CurrentKey
TRegistry class 3296
CurrentMemory
TIBDatabaseInfo class 248
CurrentPath
TRegistry class 3296
CursorInBtn
TRibbonQATScrollButton class 3771
CursorInSplit
TRibbonLargeSplitButton class 3752
TRibbonSmallSplitButton class 3796
cusDeleted enumeration member 149
cusInserted enumeration member 149
cusModified enumeration member 149
Custom
TApplyUpdatesButton class 1802
TCaptionAttributes class 1806
TCustomDataNavigator class 1823
TCustomFieldGroup class 1829
TCustomLayoutGroup class 1834
TCustomQueryButtons class 1839
TDataForm class 1849
TDataNavigator class 1857
TDataSetButton class 1865
TFieldGroup class 1873
TFieldRadioGroup class 1879
TFieldSelectOptions class 1889
TFieldStatus class 1897
TFieldText class 1902
TFieldTextArea class 1909
TGridAttributes class 1928
TGridRowAttributes class 1933

TLayoutGroup class 1941
TQueryButton class 1960
TQueryButtons class 1964
TQueryFieldGroup class 1970
TQueryForm class 1974
TQueryRadioGroup class 1982
TQuerySelectOptions class 1994
TQueryText class 2006
TQueryTextArea class 2017
TSelectOptionsColumn class 2028
TStatusColumn class 2037
TTextAreaColumn class 2045
TTextColumn class 2053
TWebButton class 2062
TWebDataDisplay class 2084
TWebForm class 2110
TWebTextInput class 2149
CustomAlignPosition
TRibbonApplicationPopupMenu class 3965
CustomFrameSupported
TCustomRibbon class 3373
CustomFrameWndProc
TCustomRibbon class 3373
Customize
TCustomRibbonQuickAccessToolbar class 3446
CustomSort
TListControlItems class 1050
cusUninserted enumeration member 149
cusUnmodified enumeration member 149

D

Data
TCustomProvider class 3115
TDataContext class 816
TIBXSQLVAR class 553
TListControlItem class 1047
TOutlineNode class 2951
TUpdateTree class 3209
Database
TIBBase class 173
TIBBlobStream class 52
TIBCustomDataSet class 99
TIBDatabaseInfo class 248
TIBEvents class 263
TIBExtract class 281
TIBSQL class 530
Database enumeration member 479
DatabaseCount
TIBTransaction class 216
DatabaseFree
TIBCustomDataSet class 110
TIBDataSet class 134
TIBQuery class 395
TIBStoredProc class 596
TIBTable class 624
DatabaseInfo
TIBExtract class 281
TIBServerProperties class 456
DatabaseName
TIBBackupService class 411
TIBConfigService class 417
TIBDataBase class 190
TIBRestoreService class 442
TIBStatisticalService class 460
TIBValidationService class 464
Databases
TIBTransaction class 216
DataContext
TRemotable class 873
DataEvent
TIBTable class 607
TPacketDataSet class 3191
DataOffset
TDataContext class 816
DataP
TDataContext class 817
DataPages enumeration member 482
DataRequest

TCustomProvider class 3101
DataSet
IDataSetComponent interface 2202
TDataSetProvider class 3160
TFieldRadioGroup class 1879
TFieldSelectOptions class 1889
TIBDataSetUpdateObject class 137
TIBUpdateSQL class 635
TQueryRadioGroup class 1982
TQuerySelectOptions class 1995
TSelectOptionsColumn class 2029
TWebListInput class 2119
TXMLDataSet class 2170
DataSetField
TXMLDataSet class 2171
DataSetPath
TXMLDataSet class 2171
DataSource
TIBDataSet class 127
TIBQuery class 388
DBConnect
IIBSQLMonitorHook interface 568
DBDisconnect
IIBSQLMonitorHook interface 569
DBFileName
TIBDatabaseInfo class 248
DBHandle
TIBBase class 173
TIBBlobStream class 52
TIBCustomDataSet class 99
TIBSQL class 530
DBImplementationClass
TIBDatabaseInfo class 249
DBImplementationNo
TIBDatabaseInfo class 249
DblClick
TCustomOutline class 2847
TCustomRibbon class 3373
TOleContainer class 2540
DbLog enumeration member 482
DbName
TDatabaseInfo class 405
DBParamByDPB
TIBDataBase class 190
DBSiteName
TIBDatabaseInfo class 249
DBSQLDialect
TIBDataBase class 190
TIBDatabaseInfo class 249
Deactivate
IBOA interface 2794
IObjectControl interface 2425
TMtsAutoObject class 2390
TMtsDataModule class 2406
DeactivateAndUndo
TOleControl class 2651
DeactivateIndexes enumeration member 480
DeactivateTransaction
TIBCustomDataSet class 73
DeclareFunctions
TBaseInetXPageProducer class 2263
DeclareIncludes
TBaseInetXPageProducer class 2263
DeclareRowSets
TBaseInetXPageProducer class 2263
DeclareScript
TBaseInetXPageProducer class 2263
DeclareStyles
TBaseInetXPageProducer class 2263
DeclareWarnings
TBaseInetXPageProducer class 2264
DeclareXMLDocuments
TBaseInetXPageProducer class 2264
DeclareXMLIslands
TBaseInetXPageProducer class 2264
Default
TMenuItem class 1406
DefaultAction

TIBTransaction class 216
DefaultCaption
TDataSetButton class 1862
TQueryButton class 1955
TXMLButton class 2156
DefaultDatabase
TIBTransaction class 217
DefaultDispatch
TOleControl class 2696
DefaultHandler
TOleControl class 2651
DefaultIndex
ITypeCode interface 2817
TIBTable class 616
DefaultTransaction
TIBDataBase class 191
DefaultWebComponents
TWebControlGroup class 2072
DefChanged
TIBTable class 607
DefineProperties
TCustomImageList class 659
TCustomOutline class 2847
TCustomRibbon class 3374
TIBQuery class 382
TIBStoredProc class 588
TMenuItem class 1375
TOleContainer class 2541
TOleControl class 2652
Delete
TCustomImageList class 659
TCustomOutline class 2848
TMenuItem class 1376
DeleteCount
TIBDatabaseInfo class 250
DeleteIndex
TIBTable class 607
DeleteKey
TCustomIniFile class 721
TIniFile class 757
TMemIniFile class 771
TRegIniFile class 3240
TRegistry class 3259
TRegistryIniFile class 3305
DeleteNode
TCustomOutline class 2848
DeleteSQL
TIBCustomDataSet class 99
TIBDataSet class 127
TIBUpdateSQL class 635
DeleteTab
TCustomRibbon class 3374
DeleteTable
TIBTable class 608
DeleteUser
TIBSecurityService class 446
DeleteValue
TRegistry class 3261
Delta
TUpdateTree class 3210
DenyAttachment enumeration member 481
DenyTransaction enumeration member 481
Desc
TLicenseInfo class 467
DesignerAdd
TRibbonTabs class 3564
Destination
TIBBatchMove class 32
DestinationDirectory
TIBInstall class 315
Destroy
TApplicationMenu class 3351
TAutomation class 2505
TAutoObject class 2508
TBaseInetXPageProducer class 2259
TBasePageItemsProducer class 2967
TBaseProvider class 3080
TChangeLink class 649

TConfigFileData class 400
TConfigParams class 402
TCustomDataGrid class 1814
TCustomFieldGroup class 1827
TCustomImageList class 654
TCustomInetXPageProducer class 2278
TCustomLayoutGroup class 1832
TCustomOutline class 2841
TCustomPacketWriter class 3094
TCustomProvider class 3099
TCustomResolver class 3127
TCustomRibbon class 3363
TCustomRibbonComboBox class 3584
TCustomRibbonGroup class 3408
TCustomRibbonPage class 3427
TCustomRibbonSkin class 4019
TCustomStaticListAction class 1029
TCustomTabData class 3455
TDatabaseInfo class 405
TDataContext class 815
TDataPacketWriter class 3137
TDataSetProvider class 3148
TEnumPropDesc class 2633
TFileStream class 2994
THashedStringList class 750
THeaderList class 824
TIBBackupService class 410
TIBBase class 167
TIBBatchMove class 28
TIBBlobStream class 42
TIBCustomDataSet class 66
TIBCustomService class 426
TIBCustomSQLMonitor class 562
TIBDataBase class 178
TIBDatabaseInfo class 234
TIBDataLink class 112
TIBDataSetUpdateObject class 136
TIBDSBlobStream class 139
TIBEvents class 260
TIBExtract class 273
TIBInputDelimitedFile class 501
TIBInputRawFile class 506
TIBInstall class 312
TIBOutputDelimitedFile class 508
TIBOutputRawFile class 512
TIBQuery class 381
TIBRestoreService class 441
TIBSecurityService class 445
TIBServerProperties class 453
TIBSQL class 516
TIBStoredProc class 587
TIBTable class 605
TIBTimer class 204
TIBTransaction class 208
TIBUpdateSQL class 631
TIBValidationService class 463
TIBXSQLDA class 537
TIniFile class 756
TInvokableClassRegistry class 842
TISAPIApplication class 961
TJPEGData class 993
TJPEGImage class 997
TKeyTipList class 3593
TLicenseInfo class 467
TLocalAppServer class 3184
TMask class 1180
TMediaPlayer class 2311
TMemIniFile class 770
TMenu class 1334
TMenuItem class 1367
TMultipartContentParser class 3343
TObjectDispatch class 2485
TOffice2007Button class 3462
TOleContainer class 2533
TOleControl class 2643
TOleForm class 2614
TOleServer class 2710
TOPToSoapDomConvert class 2751

TOutlineNode class 2943
TPageItemsProducer class 2971
TPanningWindow class 694
TPNGPointerList class 3000
TPopupMenu class 1433
TPrinter class 3040
TQueryTextArea class 2013
TQuickAccessToolbarGroupItem class 3476
TQuickAccessToolbarGroups class 3478
TRegistry class 3253
TRegistryClass class 2512
TRegistryIniFile class 3304
TRemotable class 870
TRemotableTypeRegistry class 876
TRibbonApplicationMenuBar class 3944
TRibbonApplicationMenuButton class 3605
TRibbonApplicationPopupMenu class 3962
TRibbonBaseButtonControl class 3618
TRibbonComboControl class 3686
TRibbonCustomControl class 3699
TRibbonGroupCollapsedControl class 3508
TRibbonSpinEdit class 3838
TRibbonStyleActionBars class 4023
TRibbonStylePopupMenu class 3978
TRibbonTabItem class 3555
TRIO class 4058
TSharedConnection class 1311
TSimpleObjectBroker class 2476
TSOAPAttachment class 898
TSOAPDomConv class 2761
TSQLResolver class 3198
TStringHash class 784
TTimerSpeedButton class 3908
TUpdateTree class 3207
TWebControlGroup class 2066
TWebDataDisplay class 2075
TWebDisplayReferenceGroup class 2100
TWebForm class 2104
TWebListInput class 2116
TWebRequestFile class 3334
TWebRequestFiles class 3337
TXMLButton class 2153
TXMLDataSet class 2166
TXMLDisplayGroup class 2178
TXMLDisplayReferenceButton class 2193
DestroyObject
TOleContainer class 2541
DestroyWindowHandle
TOleControl class 2652
Detach
TIBCustomService class 428
DetailCount
TUpdateTree class 3210
DetailList
TUpdateTree class 3210
Details
TUpdateTree class 3210
DeviceID
TMediaPlayer class 2338
DeviceType
TMediaPlayer class 2339
DialogAction
TCustomRibbonGroup class 3421
TRibbonGroup class 3504
DIBNeeded
TJPEGImage class 1000
Direction
TRibbonGroupScrollButton class 3524
DisableCommit
IObjectContext interface 2421
TMtsAutoObject class 2390
TMtsDataModule class 2407
DisabledFontColor
TRibbonLunaColorMap class 3528
TRibbonObsidianColorMap class 3534
TRibbonSilverColorMap class 3553
DiscardUndoState
TOleControl class 2653

Disconnect
TIBCustomDataSet class 73
TIBQuery class 383
TIBStoredProc class 588
TOleServer class 2713
DisconnectEvents
TOleServer class 2713
DiscriminatorType
ITypeCode interface 2817
Dispatch
IStub interface 2813
DispatchCommand
TMenu class 1335
DispatchPopup
TMenu class 1336
DispatchStruct
IORB interface 2807
Display
TMediaPlayer class 2339
DisplayColumns
TCustomLayoutGroup class 1834
TFieldRadioGroup class 1880
TLayoutGroup class 1941
TQueryRadioGroup class 1983
TWebRadioGroupInput class 2127
DisplayComponent
TWebDisplay class 2097
DisplayGallery
TRibbonPopupGalleryBar class 3995
DisplayKeyTips
TCustomRibbon class 3374
DisplayRect
TMediaPlayer class 2340
DisplayRows
TCustomDataGrid class 1817
TDataGrid class 1853
TFieldSelectOptions class 1890
TFieldTextArea class 1910
TQuerySelectOptions class 1995
TQueryTextArea class 2017
TSelectOptionsColumn class 2029
TTextAreaColumn class 2045
TWebSelectOptionsInput class 2132
TWebTextAreaInput class 2143
DisplayShadow
TRibbonQATCustomizeMenu class 3969
DisplayUser
TIBSecurityService class 446
DisplayUsers
TIBSecurityService class 447
DisplayWidth
TFieldRadioGroup class 1880
TFieldText class 1903
TFieldTextArea class 1910
TQueryRadioGroup class 1983
TQueryText class 2007
TQueryTextArea class 2018
TTextAreaColumn class 2046
TTextColumn class 2053
TWebRadioGroupInput class 2127
TWebStatus class 2138
TWebTextAreaInput class 2144
TWebTextInput class 2150
dkMethod enumeration member 2491
dkProperty enumeration member 2491
dkSubComponent enumeration member 2491
DoAddCustomizeItems
TRibbonQATCustomizeMenu class 3969
DoAddToQuickAccessToolbar
TCustomRibbon class 3375
DoAfterApplyUpdates
TCustomProvider class 3102
DoAfterDatabaseDisconnect
TIBBase class 171
DoAfterExecute
TCustomProvider class 3102
TRIO class 4061
DoAfterGetContent

TBaseInetXPageProducer class 2264
DoAfterGetParams
TCustomProvider class 3102
DoAfterGetRecords
TCustomProvider class 3103
DoAfterGetXMLData
TBaseInetXPageProducer class 2265
DoAfterRowRequest
TCustomProvider class 3103
DoAfterTransactionEnd
TIBBase class 171
DoAfterUpdateRecord
TBaseProvider class 3082
DoApplyUpdates
TCustomProvider class 3104
DoBeforeApplyUpdates
TCustomProvider class 3104
DoBeforeCommit
TBaseProvider class 3082
DoBeforeDatabaseDisconnect
TIBBase class 171
TIBSQL class 524
DoBeforeDelete
TIBCustomDataSet class 73
DoBeforeEdit
TIBCustomDataSet class 73
DoBeforeExecute
TCustomProvider class 3105
TDataSetProvider class 3150
TRIO class 4061
DoBeforeGetContent
TBaseInetXPageProducer class 2265
DoBeforeGetParams
TCustomProvider class 3105
DoBeforeGetRecords
TCustomProvider class 3106
TDataSetProvider class 3151
DoBeforeGetXMLData
TBaseInetXPageProducer class 2265
DoBeforeInsert
TIBCustomDataSet class 74
DoBeforeRowRequest
TCustomProvider class 3106
DoBeforeTransactionEnd
TIBBase class 171
DoBeforeUpdateRecord
TBaseProvider class 3083
DoBiDiModeChanged
TMenu class 1336
DoChange
TMenu class 1336
TQuickAccessToolbar class 3472
DoChangeQuickAccessToolbarPosition
TCustomRibbon class 3375
DockDrop
TCustomRibbonGroup class 3413
DoClick
TOffice2007Button class 3463
DoConnect
TCOMConnection class 1280
TDCOMConnection class 1287
TIBDataBase class 182
TMidasConnection class 1784
TOLEnterpriseConnection class 1307
TSharedConnection class 1311
DoContextPopup
TCustomRibbon class 3375
TCustomRibbonQuickAccessToolbar class 3446
DocumentFontColor
TCustomRibbonColorMap class 3404
TRibbonLunaColorMap class 3529
TRibbonObsidianColorMap class 3534
TRibbonSilverColorMap class 3553
DocumentName
TCustomRibbon class 3396
TRibbon class 3492
DoCustomizeQuickAccessToolbar
TCustomRibbon class 3375

DoDatabaseFree
TIBBase class 171
DoDelete
TCustomResolver class 3129
TDataSetResolver class 3179
TSQLResolver class 3199
DoDisconnect
TDispatchConnection class 1292
TIBDataBase class 183
TMidasConnection class 1785
TSharedConnection class 1312
DoDraw
TCustomImageList class 659
DoDrawText
TMenuItem class 1376
DoDropActions
TCustomRibbonQuickAccessToolbar class 3446
TRibbonApplicationPopupMenu class 3965
DoDropCategory
TCustomRibbonGalleryBar class 3984
TCustomRibbonQuickAccessToolbar class 3447
TRibbonApplicationPopupMenu class 3966
DoDropDown
TRibbonDropDownButton class 3718
TRibbonGroupDropDown class 3740
DoDropItem
TCustomRibbonGroup class 3414
TCustomRibbonQuickAccessToolbar class 3447
TRibbonApplicationPopupMenu class 3966
DoEndDrag
TCustomRibbonGroup class 3414
DoEnter
TOleContainer class 2542
DoExecSQL
TSQLResolver class 3200
DoGetControlClass
TCustomRibbonGalleryPopupBar class 3990
DoGetMenuString
TMenu class 1337
DoGetProviderAttributes
TDataSetProvider class 3152
DoGetRecords
TCustomProvider class 3107
DoGetTableName
TDataSetProvider class 3152
DoGetValues
TSQLResolver class 3200
DoGroupStyleChanged
TCustomRibbon class 3376
DoHelpButtonClick
TCustomRibbon class 3376
DoInsert
TCustomResolver class 3129
TDataSetResolver class 3179
TSQLResolver class 3200
DoLoadByFile
TCustomRibbonSkin class 4021
DoLoadByResource
TCustomRibbonSkin class 4021
DoMinimizeRibbon
TCustomRibbon class 3376
DoMoreCommands
TCustomRibbonQuickAccessToolbar class 3447
DoMouseWheelDown
TCustomRibbon class 3376
DoMouseWheelEx
TCustomRibbon class 3377
DoMouseWheelUp
TCustomRibbon class 3377
DoNotify
TMediaPlayer class 2315
DoObjectVerb
TOleControl class 2653
DoOnGetData
TBaseProvider class 3083
DoOnNewRecord
TIBCustomDataSet class 74
TIBTable class 608

DoOnUpdateData
TBaseProvider class 3083
DoPopup
TPopupMenu class 1433
DoRecentItemClick
TCustomRibbon class 3377
DoScrollBtnClick
TCustomRibbonQuickAccessToolbar class 3447
DoScrollButtonClick
TCustomRibbonPage class 3430
DoScrollTabsLeft
TCustomRibbon class 3378
DoScrollTabsRight
TCustomRibbon class 3378
DoSplitButtonClick
TRibbonLargeSplitButton class 3748
DoTabChange
TCustomRibbon class 3378
DoTagEvent
TBaseInetXPageProducer class 2265
DoTransactionFree
TIBBase class 172
DoubleBuffered
TMaskEdit class 1115
TMediaPlayer class 2340
TOutline class 2883
DoUpdate
TCustomResolver class 3129
TDataSetResolver class 3180
TPanningWindow class 696
TSQLResolver class 3201
DoUpdateQATWidth
IRibbonQuickAccessToolbar interface 3570
TCustomRibbonQuickAccessToolbar class 3448
DoUpdates
TUpdateTree class 3207
DoVerb
TOleContainer class 2542
DownClick
TRibbonSpinEdit class 3841
DownGlyph
TRibbonSpinButton class 3806
DownNumGlyphs
TRibbonSpinButton class 3806
DragCursor
TMaskEdit class 1117
TOleContainer class 2564
TOutline class 2885
TRibbonSpinButton class 3807
TRibbonSpinEdit class 3849
DragDrop
TCustomRibbonComboBox class 3586
TCustomRibbonGroup class 3414
TCustomRibbonQuickAccessToolbar class 3448
TRibbonBaseButtonControl class 3620
DragKind
TMaskEdit class 1117
TOutline class 2885
TRibbonSpinButton class 3807
DragMode
TMaskEdit class 1118
TOleContainer class 2564
TOutline class 2885
TRibbonSpinButton class 3807
TRibbonSpinEdit class 3850
DragOver
TCustomRibbonComboBox class 3586
TRibbonApplicationMenuBar class 3946
TRibbonApplicationMenuButton class 3607
Draw
TCustomImageList class 660
TJPEGImage class 1001
DrawArrow
TRibbonDropDownButton class 3718
TRibbonGroupDropDown class 3741
TRibbonLargeSplitButton class 3749
DrawArrows
TRibbonQATScrollButton class 3769

TRibbonToolScrollButton class 3906
DrawBackground
TCustomRibbonQuickAccessPopupToolbar class 3440
TCustomRibbonQuickAccessToolbar class 3448
TOffice2007Button class 3464
TRibbonAddRemoveItem class 3602
TRibbonApplicationMenuBar class 3946
TRibbonApplicationPopupMenu class 3966
TRibbonBaseButtonControl class 3621
TRibbonCheckBoxControl class 3628
TRibbonComboBoxItem class 3683
TRibbonComboControl class 3689
TRibbonCustomControl class 3702
TRibbonCustomizeItem class 3712
TRibbonCustomStatusControl class 3715
TRibbonDropDownButton class 3718
TRibbonGalleryMenuItem class 3732
TRibbonGroupDropDown class 3741
TRibbonLargeButtonControl class 3744
TRibbonLargeSplitButton class 3749
TRibbonMenuItem class 3758
TRibbonOptionButtonControl class 3766
TRibbonQATScrollButton class 3769
TRibbonRadioButtonControl class 3773
TRibbonRichContentItem class 3778
TRibbonSeparator class 3785
TRibbonSmallSplitButton class 3793
TRibbonSplitMenuItem class 3900
TRibbonStyleMenuButton class 3903
TRibbonTextControl class 3904
TRibbonToolScrollButton class 3906
DrawBanner
TRibbonMenuItem class 3760
DrawCaption
TCustomRibbon class 3378
DrawCell
TCustomOutline class 2849
DrawDown
TRibbonBaseButtonControl class 3626
TRibbonQATScrollButton class 3771
DrawElement
TRibbonStyleActionBars class 4025
DrawFrame
TRibbonComboControl class 3689
DrawGlyph
TRibbonAddRemoveItem class 3602
TRibbonApplicationMenuItem class 3614
TRibbonBaseButtonControl class 3621
TRibbonCheckBoxControl class 3628
TRibbonCustomizeItem class 3712
TRibbonGalleryMenuItem class 3732
TRibbonMenuItem class 3758
TRibbonRadioButtonControl class 3773
TRibbonRichContentItem class 3778
TRibbonRichContentMainMenuItem class 3782
TRibbonSeparator class 3786
DrawImage
THelpButton class 3460
TOffice2007Button class 3464
DrawingStyle
TCustomImageList class 681
DrawItem
TMenuItem class 1377
DrawOverlay
TCustomImageList class 663
DrawQuickAccessToolbar
TCustomRibbon class 3379
DrawSelectedBackground
TRibbonApplicationMenuItem class 3615
TRibbonMenuItem class 3758
TRibbonSplitMenuItem class 3900
DrawSeparator
TRibbonApplicationMenuItem class 3615
TRibbonBaseButtonControl class 3621
TRibbonMenuItem class 3759
TRibbonMenuSeparatorItem class 3762
DrawShadowedText
TRibbonApplicationMenuItem class 3615

TRibbonBaseButtonControl class 3622
TRibbonDropDownButton class 3718
TRibbonGalleryMenuItem class 3732
TRibbonLargeButtonControl class 3745
TRibbonMainMenuItem class 3753
TRibbonMenuItem class 3759
TRibbonRichContentItem class 3779
TRibbonSeparator class 3786
DrawShowDialogButton
TCustomRibbonGroup class 3414
DrawSubMenuGlyph
TRibbonApplicationMenuItem class 3615
DrawSystemButtons
TCustomRibbon class 3379
DrawTab
TCustomRibbon class 3379
DrawText
TRibbonApplicationMenuItem class 3616
TRibbonBaseButtonControl class 3622
TRibbonComboBoxItem class 3683
TRibbonDropDownButton class 3719
TRibbonGalleryMenuItem class 3733
TRibbonLargeButtonControl class 3745
TRibbonMainMenuItem class 3753
TRibbonMenuItem class 3759
TRibbonRichContentItem class 3779
TRibbonRichContentMainMenuItem class 3783
TRibbonSeparator class 3786
TRibbonTextControl class 3905
DropDatabase
TIBDataBase class 183
DropDownClick
TRibbonComboControl class 3690
dsFocus enumeration member 686
dsNormal enumeration member 686
dsSelected enumeration member 686
dsTransparent enumeration member 686
DSWriter
TCustomPacketWriter class 3096
dtAutoSelect enumeration member 2378
dtAVIVideo enumeration member 2378
dtCDAudio enumeration member 2378
dtDAT enumeration member 2378
dtDigitalVideo enumeration member 2378
dtMMMovie enumeration member 2378
dtOther enumeration member 2378
dtOverlay enumeration member 2378
dtScanner enumeration member 2378
dtSequencer enumeration member 2378
dtVCR enumeration member 2378
dtVideodisc enumeration member 2378
dtWaveAudio enumeration member 2378
DynArrayToClear
TDataContext class 817

E

EBrokerException class 2466
about EBrokerException class 2466
ECB
TISAPIRequest class 976
EDBEditError class 1080
about EDBEditError class 1080
EditCanModify
TCustomMaskEdit class 1084
EditMask
TCustomMaskEdit class 1092
TMaskEdit class 1119
EditorEnabled
TRibbonSpinEdit class 3851
EditText
TCustomMaskEdit class 1095
EDSWriter class 3075
about EDSWriter class 3075
Create 3076
EDSWriter Members 3076
EDSWriter properties 3076
ErrorCode 3076
EIBClientError class 3

about EIBClientError class 3
EIBError class 3
about EIBError class 3
Create 4
Create constructor 4
EIBError Members 4
EIBError properties 5
IBErrorCode 5
SQLCode 5
EIBInstall class 307
about EIBInstall class 307
Create 308
Create constructor 308
EIBInstall Members 307
EIBInstall properties 308
InstallerError 308
InstallError 309
EIBInstallerError class 309
about EIBInstallerError class 309
EIBInstallError class 310
about EIBInstallError class 310
EIBInterBaseError class 5
about EIBInterBaseError class 5
EIBInterBaseRoleError class 6
about EIBInterBaseRoleError class 6
EIniFileException class 716
about EIniFileException class 716
EInterfaceRTTIException class 791
about EInterfaceRTTIException class 791
EInvalidArgument class 1209
about EInvalidArgument class 1209
Eject
TMediaPlayer class 2315
EMaskException class 1178
about EMaskException class 1178
EMCIDeviceError class 2305
about EMCIDeviceError class 2305
EMenuError class 1321
about EMenuError class 1321
EmptyTable
TIBTable class 608
EnableCommit
IObjectContext interface 2422
TMtsAutoObject class 2391
TMtsDataModule class 2407
Enabled
IIBSQLMonitorHook interface 574
TIBCustomSQLMonitor class 563
TIBSQLMonitor class 565
TIBTimer class 205
TMaskEdit class 1122
TMediaPlayer class 2342
TMenuItem class 1406
TOleContainer class 2566
TOutline class 2887
TRibbon class 3492
TRibbonSpinButton class 3809
TRibbonSpinEdit class 3851
TServerItem class 2473
TStaticListAction class 1057
TVirtualListAction class 1068
EnabledButtons
TMediaPlayer class 2343
EnableModeless
TOleControl class 2653
EnableValidateFields
IValidateFields interface 2226
TXMLDisplayGroup class 2186
EnableXMLIslands
TBaseInetXPageProducer class 2271
TInetXPageProducer class 2288
Encoding
IOPConvert interface 2738
TMemIniFile class 782
TOPToSoapDomConvert class 2756
TSOAPAttachment class 906
EndButtons
TFormLayout class 1917

EndDoc
TPrinter class 3045
EndFields
TFormLayout class 1917
EndLayout
TFormLayout class 1917
EndPos
TMediaPlayer class 2345
EndRow
TFormLayout class 1918
EndTable
TFormLayout class 1918
EndUpdate
TCustomOutline class 2849
TCustomResolver class 3130
TCustomRibbon class 3379
TDataSetResolver class 3180
EnsureBlobInitialized
TIBBlobStream class 44
EnumNames
IGetContextProperties interface 2420
eoBLOBFilter enumeration member 283
eoChecks enumeration member 283
eoData enumeration member 283
eoDatabase enumeration member 283
eoDomain enumeration member 283
eoException enumeration member 283
Eof
TIBControlAndQueryService class 421
TIBSQL class 530
eoForeign enumeration member 283
eoFunction enumeration member 283
eoGenerator enumeration member 283
eoIndexes enumeration member 283
EOleCtrlError class 2631
about EOleCtrlError class 2631
EOleError class 2497
about EOleError class 2497
EOleException class 2497
about EOleException class 2497
Create 2498
EOleException Members 2498
EOleException properties 2498
ErrorCode 2498
HelpFile 2499
Source 2499
EOleSysError class 2499
about EOleSysError class 2499
Create 2500
EOleSysError Members 2499
EOleSysError properties 2500
ErrorCode 2500
eoProcedure enumeration member 283
eoRole enumeration member 283
eoTable enumeration member 283
eoTrigger enumeration member 283
EOutlineError class 2837
about EOutlineError class 2837
eoView enumeration member 283
EPNGCannotChangeTransparent class 2984
about EPNGCannotChangeTransparent class 2984
EPNGCouldNotLoadResource class 2985
about EPNGCouldNotLoadResource class 2985
EPngError class 2985
about EPngError class 2985
EPNGHeaderNotPresent class 2985
about EPNGHeaderNotPresent class 2985
EPNGIHDRNotFirst class 2986
about EPNGIHDRNotFirst class 2986
EPngInvalidCRC class 2986
about EPngInvalidCRC class 2986
EPNGInvalidFileHeader class 2987
about EPNGInvalidFileHeader class 2987
EPngInvalidIHDR class 2987
about EPngInvalidIHDR class 2987
EPNGInvalidNewSize class 2987
about EPNGInvalidNewSize class 2987
EPNGInvalidPalette class 2988

about EPNGInvalidPalette class 2988
EPNGInvalidSpec class 2988
about EPNGInvalidSpec class 2988
EPNGMissingMultipleIDAT class 2988
about EPNGMissingMultipleIDAT class 2988
EPNGMissingPalette class 2989
about EPNGMissingPalette class 2989
EPNGNoImageData class 2989
about EPNGNoImageData class 2989
EPNGNotExists class 2990
about EPNGNotExists class 2990
EPNGOutMemory class 2990
about EPNGOutMemory class 2990
EPNGSizeExceeds class 2990
about EPNGSizeExceeds class 2990
EPngUnexpectedEnd class 2991
about EPngUnexpectedEnd class 2991
EPNGUnknownCompression class 2991
about EPNGUnknownCompression class 2991
EPNGUnknownCriticalChunk class 2991
about EPNGUnknownCriticalChunk class 2991
EPNGUnknownInterlace class 2992
about EPNGUnknownInterlace class 2992
EPNGZLIBError class 2992
about EPNGZLIBError class 2992
EPrinter class 3038
about EPrinter class 3038
Equal
ITypeCode interface 2818
Equals
TJPEGImage class 1001
erAbort enumeration member 329
EraseSection
TCustomIniFile class 722
TIniFile class 757
TMemIniFile class 772
TRegIniFile class 3241
TRegistryIniFile class 3306
erContinue enumeration member 329
ERegistryException class 3236
about ERegistryException class 3236
ERemotableException class 810
about ERemotableException class 810
Create 811
Create constructor 811
ERemotableException Members 811
ERemotableException properties 812
FaultActor 812
FaultCode 812
FaultDetail 813
ERibbonException class 3349
about ERibbonException class 3349
erRetry enumeration member 329
Error
TMediaPlayer class 2346
ErrorCode
EDSWriter class 3076
EOleException class 2498
EOleSysError class 2500
SQLException class 989
ErrorContext
TIBSetup class 319
ErrorDS
TUpdateTree class 3211
ErrorInternal
TIBSetup class 318
ErrorMessage
TMediaPlayer class 2346
ESOAPDomConvertError class 2747
about ESOAPDomConvertError class 2747
etAlterProc enumeration member 284
etCheck enumeration member 284
etData enumeration member 284
etDomain enumeration member 284
etForeign enumeration member 284
etGrant enumeration member 284
etIndex enumeration member 284
etRole enumeration member 284

etTable enumeration member 284
etTrigger enumeration member 284
ETypeRegistryException class 813
about ETypeRegistryException class 813
EventChange
TIBEvents class 260
EventContent
TQueryButton class 1956
TResetQueryButton class 2024
TSubmitQueryButton class 2040
TWebDataDisplay class 2077
TWebDataInput class 2089
TWebRadioGroupInput class 2126
TWebSelectOptionsInput class 2132
TWebTextAreaInput class 2142
EventDispatch
TOleServer class 2718
Events
TIBEvents class 264
Examples
TInstallOptions class 326
exAPI enumeration member 330
Exception class 2993
about Exception class 2993
Create 2993
Exception Members 2993
exDB enumeration member 330
exDenormalized enumeration member 1263
ExecProc
TIBStoredProc class 588
ExecQuery
TIBSQL class 524
ExecSQL
TIBDataSet class 119
TIBQuery class 383
TIBUpdateSQL class 633
Execute
IMethodHandler interface 2452
ISkeletonObject interface 2811
TCustomProvider class 3107
TIBBatchMove class 28
ExecuteTarget
TCustomListAction class 1021
exInvalidOp enumeration member 1263
Exists
TIBTable class 616
TQuickAccessToolbarGroups class 3479
exOverflow enumeration member 1263
Expand
TCustomOutline class 2850
TOutlineNode class 2945
ExpandCount
TOutlineNode class 2952
Expanded
TOutlineNode class 2952
Exported
TCustomProvider class 3115
TDataSetProvider class 3160
exPrecision enumeration member 1263
ExpungeCount
TIBDatabaseInfo class 250
Extract
THeaderList class 826
ExtractContentTypeFields
TMultipartContentParser class 3344
ExtractDDL
TIBExtract class 274
ExtractListTable
TIBExtract class 274
ExtractListView
TIBExtract class 274
ExtractMultipartContentFields
TMultipartContentParser class 3345
ExtractObject
TIBExtract class 274
exUnderflow enumeration member 1263
exZeroDivide enumeration member 1263

F

FAfterDatabaseDisconnect
TIBBase class 167
FAfterTransactionEnd
TIBBase class 167
FaultActor
ERemotableException class 812
FaultCode
ERemotableException class 812
FaultDetail
ERemotableException class 813
FBackoutCount
TIBDatabaseInfo class 235
FBase
TIBSQL class 517
FBeforeDatabaseDisconnect
TIBBase class 168
FBeforeTransactionEnd
TIBBase class 168
FBlobStream
TIBDSBlobStream class 139
FBOF
TIBSQL class 517
FBuffer
TCustomPacketWriter class 3095
FCaption
TListControlItem class 1043
FClient
TMenuActionLink class 1354
FColDelimiter
TIBInputDelimitedFile class 501
TIBOutputDelimitedFile class 509
FColumns
TIBBatch class 495
FConverter
TRIO class 4059
FCount
TIBXSQLDA class 538
FCursor
TIBSQL class 517
FCursorInSplit
TRibbonSmallSplitButton class 3791
FData
TListControlItem class 1043
FDatabase
TIBBase class 168
TIBDatabaseInfo class 235
FDeleteCount
TIBDatabaseInfo class 236
FEOF
TIBInputDelimitedFile class 501
TIBSQL class 517
Fetch
TIBServerProperties class 454
FetchAll
TIBCustomDataSet class 74
FetchConfigParams
TIBServerProperties class 454
FetchDatabaseInfo
TIBServerProperties class 455
FetchDetails
TBaseProvider class 3083
TDataSetProvider class 3153
Fetches
TIBDatabaseInfo class 250
FetchLicenseInfo
TIBServerProperties class 455
FetchLicenseMaskInfo
TIBServerProperties class 455
FetchLimboTransactionInfo
TIBValidationService class 463
FetchParams
IInetXWebDataSet interface 2215
FetchVersionInfo
TIBServerProperties class 455
FExpungeCount
TIBDatabaseInfo class 236

FField
TIBDSBlobStream class 140
FFile
TIBInputDelimitedFile class 502
TIBInputRawFile class 506
TIBOutputDelimitedFile class 509
TIBOutputRawFile class 512
FFilename
TIBBatch class 495
FGenerateParamNames
TIBSQL class 518
FGoToFirstRecordOnExecute
TIBSQL class 518
FHandle
TIBSQL class 518
FHeadersInbound
TSOAPHeadersBase class 917
FHeadersOutBound
TSOAPHeadersBase class 918
FIBLoaded
TIBDatabaseInfo class 236
Field
TIBGeneratorField class 145
FieldByName
TIBSQL class 524
FieldCount
IInetXWebDataSet interface 2215
FieldDefs
TIBTable class 617
FieldIndex
TIBSQL class 531
FieldName
IDataSetField interface 2205
TFieldRadioGroup class 1880
TFieldSelectOptions class 1890
TFieldText class 1903
TFieldTextArea class 1910
TQueryRadioGroup class 1984
TQuerySelectOptions class 1996
TQueryText class 2007
TQueryTextArea class 2018
TSelectOptionsColumn class 2030
TTextAreaColumn class 2046
TTextColumn class 2053
TWebDataInput class 2094
Fields
IInetXWebDataSet interface 2216
TIBSQL class 531
FIID
TRIO class 4059
FileLoad
TCustomImageList class 664
Filename
TIBBatch class 497
FileName
TCustomIniFile class 749
TMediaPlayer class 2347
TRegIniFile class 3246
Filter
TIBTable class 617
Filtered
TIBDataSet class 127
TIBQuery class 389
TIBStoredProc class 592
TIBTable class 617
FImageIndex
TListControlItem class 1043
Finalize
TIBBlobStream class 45
FinalizeMultiRefNodes
TSOAPDomConv class 2769
Find
TMenuItem class 1377
TStringHash class 785
FindAssociatedField
TWebDataInput class 2090
FindAttachment
IObjConverter interface 922

TSOAPDomConv class 2769
FindChunk
TPNGList class 2998
FindDatabase
TIBTransaction class 211
FindDataSet
TXMLDataSet class 2168
FindDefaultDatabase
TIBTransaction class 212
FindDefaultTransaction
TIBDataBase class 183
FIndexInDatabase
TIBBase class 168
FIndexInTransaction
TIBBase class 169
FindField
TXMLDisplayGroup class 2180
FindGroup
TCustomRibbonPage class 3430
TRibbonTabs class 3562
FindGroupAt
TCustomRibbonPage class 3430
FindItem
TMenu class 1337
FindMultiRefNodeByID
TSOAPDomConv class 2770
FindMultiRefNodeByInstance
TSOAPDomConv class 2770
FindNodeByHREF
TSOAPDomConv class 2770
FindParam
TQueryFieldGroup class 1969
FindPrefixForURI
TSOAPDomConv class 2770
FindRecord
TDataSetProvider class 3153
FindRepositoryTC
IORB interface 2808
FindServer
TServerCollection class 2468
FindTransaction
TIBDataBase class 184
FindXMLBroker
TBaseInetXPageProducer class 2265
FindXMLBrokerName
TBaseInetXPageProducer class 2266
FInsertCount
TIBDatabaseInfo class 236
FirstName
TIBSecurityService class 449
TUserInfo class 474
FirstVisibleTabIndex
TCustomTabData class 3456
FixLimboTransactionErrors
TIBValidationService class 463
fkCommand enumeration member 1460
fkHandle enumeration member 1460
fkShortCut enumeration member 1460
FLookAhead
TIBInputDelimitedFile class 502
FlushSchema
TIBDataBase class 184
fmGet enumeration member 2238
fmoAllBaseMethods enumeration member 797
FModified
TIBDSBlobStream class 140
fmoNoBaseMethods enumeration member 797
fmoRTTIBaseMethods enumeration member 797
FMouseInSplit
TRibbonSplitMenuItem class 3898
fmPost enumeration member 2238
FNames
TIBXSQLDA class 538
FObjsToDestroy
TDataContext class 817
FocusControl
TRibbonSpinButton class 3809
FOnDatabaseFree

TIBBase class 169
FOnSQLChanging
TIBSQL class 518
Font
TMaskEdit class 1123
TOutline class 2887
TRibbon class 3492
TRibbonComboBox class 3632
TRibbonSpinEdit class 3852
FontColor
TRibbonLunaColorMap class 3529
TRibbonObsidianColorMap class 3535
TRibbonSilverColorMap class 3553
FOnTransactionFree
TIBBase class 169
Fonts
TPrinter class 3054
FOpen
TIBSQL class 519
ForceClose
TIBDataBase class 184
Forced enumeration member 481
ForcedRefresh
TIBCustomDataSet class 99
ForcedWrites
TIBDatabaseInfo class 250
Form
IVCLFrameForm interface 2616
FormatCaption
TWebDataDisplay class 2077
FormatColumnData
IFormatColumn interface 2209
FormatColumnHeading
IFormatColumn interface 2209
FormatInputs
TWebListInput class 2117
TWebRadioGroupInput class 2126
TWebSelectOptionsInput class 2132
FormatTable
TCustomDataGrid class 1814
FormsContent
TInetXPageElements class 2284
FOutputTitles
TIBOutputDelimitedFile class 509
FOwner
TIBBase class 169
FParamCheck
TIBSQL class 519
FParams
TIBBatch class 496
FPrepared
TIBSQL class 519
FProcessedSQL
TIBSQL class 519
FPurgeCount
TIBDatabaseInfo class 236
Frames
TMediaPlayer class 2349
FReadBlanksAsNull
TIBInputDelimitedFile class 502
FReadIdxCount
TIBDatabaseInfo class 237
FReadSeqCount
TIBDatabaseInfo class 237
FRecordCount
TIBSQL class 519
FreeBitmap
TJPEGImage class 1001
FreeHandle
TIBSQL class 524
TJPEGData class 993
FreeInfoRecords
TDataPacketWriter class 3140
FreeNodes
TIBSchema class 202
FreeRecordBuffer
TIBCustomDataSet class 75
FreeResolver

TBaseProvider class 3084
FreeTreeData
TCustomResolver class 3130
TSQLResolver class 3201
FRefCount
TInvokableClass class 836
FRowDelimiter
TIBInputDelimitedFile class 502
TIBOutputDelimitedFile class 509
FSerializationOptions
TRemotable class 871
FSize
TIBXSQLDA class 538
FSkipTitles
TIBInputDelimitedFile class 503
fsmCreate enumeration member 3020
fsmRead enumeration member 3020
fsmWrite enumeration member 3020
FSOAPHeaders
TInvokableClass class 836
FSplitRect
TRibbonSmallSplitButton class 3791
TRibbonSplitMenuItem class 3899
FSQL
TIBSQL class 520
TIBXSQLDA class 539
FSQLParams
TIBSQL class 520
FSQLRecord
TIBSQL class 520
FSQLType
TIBSQL class 520
FTransaction
TIBBase class 169
Full
TKeyTipList class 3595
FullCollapse
TCustomOutline class 2850
FullExpand
TCustomOutline class 2850
TOutlineNode class 2945
FullPath
TOutlineNode class 2952
FullWidth
TCustomTabData class 3457
FUniqueRelationName
TIBXSQLDA class 539
FUpdateCount
TIBDatabaseInfo class 237
FUserNames
TIBDatabaseInfo class 237
FWebNode
TRIO class 4059
FWindow
TPopupList class 1428
FXSQLDA
TIBXSQLDA class 539
FXSQLVARs
TIBXSQLDA class 539

G

gaHorizontal enumeration member 3408
Gallery
TRibbonGalleryControl class 3722
GalleryResize
TRibbonGripItem class 3738
TRibbonStylePopupMenu class 3980
gamOnNewRecord enumeration member 151
gamOnPost enumeration member 151
gamOnServer enumeration member 151
gaVertical enumeration member 3408
GenDeleteSQL
TSQLResolver class 3201
GeneratePageElements
TBaseInetXPageProducer class 2266
GenerateParamNames
TIBQuery class 389
TIBSQL class 531

GenerateQueryForLiveUpdate
TIBQuery class 383
Generator
TIBGeneratorField class 145
GeneratorField
TIBCustomDataSet class 100
TIBDataSet class 127
TIBQuery class 389
GenInsertSQL
TSQLResolver class 3202
GenSelectSQL
TSQLResolver class 3202
GenUpdateSQL
TSQLResolver class 3202
GenWhereSQL
TSQLResolver class 3202
Get
ISOAPHeaders interface 925
TSOAPHeaders class 914
Get__NewEnum
ISharedPropertyGroupManager interface 2432
Get_Group
ISharedPropertyGroupManager interface 2433
Get_PropertyByName
ISharedPropertyGroup interface 2430
Get_PropertyByPosition
ISharedPropertyGroup interface 2430
Get_Value
ISharedProperty interface 2428
GetActionClientItem
TApplicationMenu class 3352
GetActionLinkClass
TMenuItem class 1377
GetActionURIOfIID
TInvokableClassRegistry class 844
GetActionURIOfInfo
TInvokableClassRegistry class 844
GetActiveBuf
TIBCustomDataSet class 75
GetActivityId
IObjectContextActivity interface 2424
GetAddRemoveItemClass
TPlatformDefaultStyleActionBars class 2977
TRibbonStyleActionBars class 4026
GetAllocation
TIBDatabaseInfo class 239
GetAny
IMarshalInBuffer interface 2798
GetArrayField
TIBExtract class 276
GetArrowPosition
TRibbonDropDownButton class 3719
TRibbonLargeSplitButton class 3749
TRibbonSmallSplitButton class 3794
GetAsCurrency
TIBBCDField class 61
GetAsString
TIBBCDField class 61
TIBStringField class 148
GetAsVariant
TIBBCDField class 61
TIBStringField class 148
GetAttachments
IOPConvert interface 2732
GetAutoConnect
TOleServer class 2714
GetAutoHotkeys
TMenuItem class 1378
GetAutoLineReduction
TMenuItem class 1378
GetAutoObject
TAutoDispatch class 2502
GetAutoRegister
IIBEventNotifier interface 222
TIBEvents class 260
GetBackoutCount
TIBDatabaseInfo class 239
GetBalancedName

TServerCollection class 2468
GetBarHeight
TCustomRibbonQuickAccessToolbar class 3448
GetBarWidth
TCustomRibbonQuickAccessToolbar class 3449
GetBaseKey
TRegistry class 3262
GetBaseLevel
TIBDatabaseInfo class 239
GetBitmap
TCustomImageList class 664
GetBlobInfo
TIBBlobStream class 45
GetBookmarkData
TIBCustomDataSet class 75
GetBookmarkFlag
TIBCustomDataSet class 75
GetBorder
TOleControl class 2654
GetBorderHeight
TRibbonApplicationPopupMenu class 3966
GetByteProp
TOleControl class 2654
GetCanModify
TIBCustomDataSet class 75
TIBTable class 608
GetCaption
TDataSetButton class 1862
TQueryButton class 1956
TWebButton class 2060
TWebDataDisplay class 2078
TWebDataInput class 2090
TWebStatus class 2136
TXMLButton class 2154
GetCaptionHeight
TCustomRibbonGroup class 3414
GetCaptionWidth
TRibbonComboControl class 3690
TRibbonCustomControl class 3702
GetChar
IMarshalInBuffer interface 2798
GetCharacterSets
TIBExtract class 276
GetCheckIndex
TQueryRadioGroup class 1979
TWebRadioGroupInput class 2126
GetChildren
TBasePageItemsProducer class 2968
TMenu class 1338
TMenuItem class 1378
TRibbonSpinEdit class 3841
GetClassFromIntfInfo
TInvokableClassRegistry class 845
GetColorMapClass
TPlatformDefaultStyleActionBars class 2977
TRibbonLunaStyleActionBars class 4000
TRibbonObsidianStyleActionBars class 4006
TRibbonSilverStyleActionBars class 4012
TRibbonStyleActionBars class 4026
GetColorProp
TOleControl class 2655
GetColumn
TIBInputDelimitedFile class 503
GetComponentList
TBasePageItemsProducer class 2968
GetCompProp
TOleControl class 2655
GetComputerForGUID
TCustomObjectBroker class 1282
TSimpleObjectBroker class 2477
GetComputerForProgID
TCustomObjectBroker class 1283
TSimpleObjectBroker class 2477
GetConnected
TDispatchConnection class 1293
TIBDataBase class 184
TSharedConnection class 1312
GetConnectKind

TOleServer class 2714
GetContainedControl
IRibbonContainedControl interface 3915
TRibbonComboControl class 3690
TRibbonCustomControl class 3702
TRibbonGalleryControl class 3721
TRibbonGalleryControlMenuItem class 3724
GetContainer
TOleControl class 2655
GetContent
TISAPIResponse class 979
GetContentFields
TMultipartContentParser class 3345
GetContentOptions
TBaseInetXPageProducer class 2267
GetContentType
TWebRequestFile class 3334
GetContextMenuOptions
TCustomRibbon class 3379
GetControlClass
TCustomActionControlBar class 3357
TCustomRibbonQuickAccessPopupToolbar class 3441
TPlatformDefaultStyleActionBars class 2977
TRibbonApplicationMenuBar class 3946
TRibbonQATCustomizeMenu class 3969
TRibbonStyleActionBars class 4027
GetCount
TCustomListAction class 1021
TCustomStaticListAction class 1030
THeaderList class 826
TMenuItem class 1378
TWebRequestFiles class 3338
GetCurrencyProp
TOleControl class 2656
GetCurrent
TKeyTipListEnumerator class 3597
TMenuItemEnumerator class 1425
GetCurrentMemory
TIBDatabaseInfo class 239
GetCurrentRecord
TIBCustomDataSet class 76
GetData
TRegistry class 3262
GetDataAsString
TRegistry class 3262
GetDatabase
TIBBlobStream class 45
TIBSQL class 525
GetDataInfo
TRegistry class 3263
GetDataItem
TCustomOutline class 2851
TOutlineNode class 2946
GetDataNode
TSOAPDomConv class 2771
GetDataPacket
TDataPacketWriter class 3140
GetDataPointer
TDataContext class 822
GetDataSet
IDataSetComponent interface 2201
TIBDataSetUpdateObject class 136
TIBUpdateSQL class 633
TXMLDataSet class 2168
TXMLDisplayGroup class 2180
GetDataSetField
TXMLDataSet class 2169
TXMLDataSetParent class 2174
TXMLDisplayGroup class 2180
GetDataSetFromDelta
TDataSetProvider class 3153
GetDataSetPath
TXMLDataSet class 2169
GetDataSize
TIBBCDField class 61
TRegistry class 3263
GetDataSource
TIBCustomDataSet class 76

TIBTable class 609
GetDataType
TRegistry class 3264
GetDateVariable
TISAPIRequest class 972
TISAPIResponse class 979
GetDBFileName
TIBDatabaseInfo class 239
GetDBHandle
TIBBase class 172
TIBBlobStream class 45
TIBSQL class 525
GetDBImplementationClass
TIBDatabaseInfo class 240
GetDBImplementationNo
TIBDatabaseInfo class 240
GetDBSiteName
TIBDatabaseInfo class 240
GetDBSQLDialect
TIBDatabaseInfo class 240
GetDefaultButtons
TCustomDataNavigator class 1821
TCustomQueryButtons class 1837
GetDefaultComponentList
TBasePageItemsProducer class 2969
GetDefaultTemplate
TBaseInetXPageProducer class 2267
GetDefaultWebComponents
TWebControlGroup class 2068
TWebForm class 2106
GetDeleteCount
TIBDatabaseInfo class 241
GetDetailDataSet
TIBDataLink class 113
GetDetailLinkFields
TIBQuery class 383
TIBTable class 609
GetDirectCallerSID
ISecurityProperty interface 2426
GetDirectCreatorSID
ISecurityProperty interface 2426
GetDisplayComponent
TWebDisplay class 2096
TXMLDisplayParent class 2190
TXMLDisplayReferenceButton class 2194
GetDisplayComponentParent
TXMLDisplayReferenceButton class 2194
GetDisplayName
TListControlItem class 1045
TOptionItem class 3936
TRibbonTabItem class 3557
TServerItem class 2472
GetDisplayWidth
TOutlineNode class 2946
GetDouble
IMarshalInBuffer interface 2798
GetDoubleProp
TOleControl class 2656
GetElementType
TSOAPDomConv class 2771
GetEmpty
TJPEGImage class 1001
GetEnabled
IIBSQLMonitorHook interface 569
GetEnableValidateFields
IValidateFields interface 2225
TXMLDisplayGroup class 2180
GetEncoding
IOPConvert interface 2732
GetEnumerator
TMenuItem class 1379
GetEnumPropDesc
TOleControl class 2656
GetEOF
TIBSQL class 525
GetExceptionInfo
TAutoObject class 2510
GetExceptionInfoForInterface

TInvokableClassRegistry class 845
GetExpungeCount
TIBDatabaseInfo class 241
GetExtendedControl
TOleControl class 2657
GetExtensionVersion
TISAPIApplication class 962
GetExternalPropName
TRemotableTypeRegistry class 879
GetFetches
TIBDatabaseInfo class 241
GetFieldByName
TISAPIRequest class 972
GetFieldClass
TIBCustomDataSet class 76
GetFieldData
TIBCustomDataSet class 77
GetFieldIdx
TDataPacketWriter class 3140
GetFieldIndex
TIBSQL class 525
GetFieldName
IDataSetField interface 2204
TWebDataInput class 2090
TWebRequestFile class 3334
GetFieldNames
TIBDataBase class 185
GetFields
TIBSQL class 525
GetFieldsList
IDataSetFields interface 2207
TXMLDisplayGroup class 2181
GetFieldType
TIBExtract class 276
GetFileName
TWebRequestFile class 3335
GetFiles
TMultipartContentParser class 3345
getFirstChild
TOutlineNode class 2946
GetFirstEditChar
TCustomMaskEdit class 1084
GetFloat
IMarshalInBuffer interface 2798
GetForcedWrites
TIBDatabaseInfo class 241
GetGlyphWidth
TRibbonComboControl class 3690
TRibbonCustomControl class 3702
GetGroupHeight
TCustomRibbonGroup class 3415
GetGroupPos
TCustomRibbonPage class 3431
GetHandle
TMainMenu class 1324
TMenu class 1338
TMenuItem class 1379
GetHeader
THeaderList class 827
GetHeaderClass
TInvokableClassRegistry class 846
GetHeaderInfoForInterface
TInvokableClassRegistry class 846
GetHeaderName
TInvokableClassRegistry class 847
GetHeaderNamespace
TInvokableClassRegistry class 848
GetHeight
TJPEGImage class 1002
GetHelpButtonRect
TCustomRibbon class 3380
GetHelpContext
TMenu class 1338
TOleControl class 2657
GetHotSpot
TCustomImageList class 665
GetHTMLControlName
IHTMLField interface 2210

TQueryButton class 1957
TWebDataDisplay class 2078
GetHTMLForm
TQueryButton class 1957
TWebDataDisplay class 2078
GetHTMLFormName
IHTMLForm interface 2212
TWebForm class 2107
GetHTMLFormTag
TCustomDataForm class 1810
TCustomQueryForm class 1844
GetHTMLFormVarName
IHTMLForm interface 2212
TWebForm class 2107
GetIcon
TCustomImageList class 665
TCustomRibbon class 3380
GetIconMetaPict
TOleContainer class 2542
GetIDispatchProp
TOleControl class 2657
GetIDsOfNames
TAutoDispatch class 2502
TEventDispatch class 2636
TLocalAppServer class 3186
TObjectDispatch class 2486
TOleControl class 2658
TServerEventDispatch class 2721
GetImageBitmap
TCustomImageList class 666
GetImageList
TMenuItem class 1379
GetImages
TCustomImageList class 666
GetImageSize
TRibbonApplicationMenuItem class 3616
TRibbonBaseButtonControl class 3622
TRibbonRichContentMainMenuItem class 3783
GetImplementation
ISkeleton interface 2810
ISkeletonObject interface 2811
GetIndexNames
TIBTable class 609
GetIndexParams
TIBTable class 609
GetInfoForURI
TInvokableClassRegistry class 848
GetInputType
TQueryButton class 1957
GetInsertCount
TIBDatabaseInfo class 241
GetInstallMessage
TIBSetup class 319
GetInstRes
TCustomImageList class 667
GetIntegerProp
TOleControl class 2658
GetIntegerVariable
TISAPIRequest class 973
TISAPIResponse class 980
GetInterface
IStub interface 2813
GetInterfaceCount
TInvokableClassRegistry class 849
GetInterfaceExternalName
TInvokableClassRegistry class 849
GetInterfaceInfoFromName
TInvokableClassRegistry class 850
GetInterfaceTypeInfo
TInvokableClassRegistry class 850
GetInternalPropName
TRemotableTypeRegistry class 879
GetIntfInvokeOptions
TInvokableClassRegistry class 851
GetInvokableObjectFromClass
TInvokableClassRegistry class 852
GetIsMultipleRecordView
IWebDisplay interface 2227

IXMLDisplay interface 2229
TXMLDisplayGroup class 2181
GetIsPanning
TPanningWindow class 696
GetItem
TCustomOutline class 2851
TCustomStaticListAction class 1030
TCustomVirtualListAction class 1035
TMenuItem class 1380
TWebRequestFiles class 3339
GetItemClass
TCustomStaticListAction class 1030
GetItemHeight
TRibbonRichContentItem class 3779
GetItemWidth
TRibbonRichContentItem class 3780
GetIUnknownProp
TOleControl class 2658
GetKey
TRegistry class 3264
GetKeyInfo
TRegistry class 3264
GetKeyNames
TRegistry class 3265
GetKeyTip
IRibbonKeyTip interface 3919
TCustomRibbonGroup class 3415
TRibbonApplicationMenuButton class 3607
TRibbonBaseButtonControl class 3622
TRibbonComboControl class 3690
TRibbonCustomControl class 3703
TRibbonGroupCollapsedControl class 3510
GetLabelWidth
TRibbonComboControl class 3691
TRibbonCustomControl class 3703
GetLastChild
TOutlineNode class 2947
GetLastEditChar
TCustomMaskEdit class 1085
GetLayoutAttributes
TCustomFieldGroup class 1828
TCustomLayoutGroup class 1832
TWebButton class 2060
TWebControlGroup class 2069
TWebDataDisplay class 2079
TWebForm class 2107
GetLoadType
TCustomRibbonSkin class 4021
TRibbonLunaSkin class 3999
TRibbonObsidianSkin class 4005
TRibbonSilverSkin class 4011
GetLoadTypeValue
TCustomRibbonSkin class 4021
TRibbonLunaSkin class 3999
TRibbonObsidianSkin class 4005
TRibbonSilverSkin class 4011
GetLogMessage
TISAPIResponse class 980
GetLong
IMarshalInBuffer interface 2798
GetLongDatabaseInfo
TIBDatabaseInfo class 242
GetMarks
TIBDatabaseInfo class 242
GetMaskBitmap
TCustomImageList class 668
GetMaxChars
TCustomMaskEdit class 1085
GetMaxMemory
TIBDatabaseInfo class 242
GetMenuIndex
TMenuItem class 1380
GetMenuProps
TRibbonRichContentItem class 3780
GetMethExternalName
TInvokableClassRegistry class 852
GetMethInternalName
TInvokableClassRegistry class 852

GetMethodInfo
TObjectDispatch class 2486
GetMethodName
TDataSetButton class 1863
GetModified
TIBXSQLDA class 540
GetMoniker
TOleControl class 2659
GetMonitorCount
IIBSQLMonitorHook interface 569
GetNames
TIBXSQLDA class 541
GetNamespaceByGUID
TInvokableClassRegistry class 853
GetNativeHandle
TIBEvents class 261
GetNewID
TSOAPDomConv class 2771
GetNextChild
TOutlineNode class 2947
GetNextChunk
TIBControlAndQueryService class 419
GetNextEditChar
TCustomMaskEdit class 1086
GetNextLine
TIBControlAndQueryService class 420
GetNextName
TServerCollection class 2469
GetNextVisibleGroup
TCustomRibbonPage class 3431
GetNodeAsText
TSOAPDomConv class 2771
GetNodeAtIndex
TOutlineNode class 2947
GetNodeDisplayWidth
TCustomOutline class 2852
GetNoReserve
TIBDatabaseInfo class 242
GetNumBuffers
TIBDatabaseInfo class 243
GetObject
IMarshalInBuffer interface 2799
GetObjectDispatch
TObjectDispatch class 2486
GetObjectPropAsText
TSOAPDomConv class 2771
GetObjectVerbs
TOleControl class 2659
GetODSMajorVersion
TIBDatabaseInfo class 243
GetODSMinorVersion
TIBDatabaseInfo class 243
GetOle2AcceleratorTable
TMainMenu class 1324
GetOleBoolProp
TOleControl class 2659
GetOleDateProp
TOleControl class 2660
GetOleEnumProp
TOleControl class 2660
GetOleVariantProp
TOleControl class 2660
GetOperationCounts
TIBDatabaseInfo class 243
GetOptionDescription
TIBInstall class 312
GetOptionName
TIBInstall class 313
GetOptions
IOPConvert interface 2732
TSOAPDomConv class 2772
GetOptionSpaceRequired
TIBInstall class 314
GetOriginalCallerSID
ISecurityProperty interface 2426
GetOriginalCreatorSID
ISecurityProperty interface 2427
GetOwnsObjects

THeaderList class 827
GetOwnsSentHeaders
ISOAPHeaders interface 925
TSOAPHeaders class 914
GetPageFromPageCaption
TRibbonTabs class 3562
GetPageHeight
TCustomRibbonPage class 3431
GetPageSize
TIBDatabaseInfo class 244
GetPageTop
TCustomRibbonPage class 3431
GetPalette
TJPEGImage class 1002
GetPanOptions
TPanningWindow class 696
GetParamExternalName
TInvokableClassRegistry class 853
GetParamInternalName
TInvokableClassRegistry class 854
GetParamName
IQueryField interface 2218
TWebDataInput class 2090
GetParamPointer
TInvContext class 831
GetParams
TCustomProvider class 3108
GetParamsCount
TIBQuery class 384
TIBStoredProc class 589
GetParamsList
IQueryFields interface 2220
TQueryFieldGroup class 1969
GetParentComponent
TMenuItem class 1380
GetParentControl
IRibbonGroupDropdown interface 3918
TRibbonGroupCollapsedControl class 3511
TRibbonGroupDropDown class 3741
GetParentMenu
TMenuItem class 1380
GetPlan
TIBSQL class 526
GetPopupClass
TPlatformDefaultStyleActionBars class 2978
TRibbonDropDownButton class 3719
TRibbonGallerySplitButton class 3735
TRibbonStyleActionBars class 4027
GetPopupMenu
TOleContainer class 2543
GetPortForComputer
TCustomObjectBroker class 1283
TSimpleObjectBroker class 2478
GetPosition
TStream class 3006
GetPrevChild
TOutlineNode class 2948
GetPrincipal
IBOA interface 2794
GetPrincipalLength
IBOA interface 2794
GetPrinter
TPrinter class 3047
GetPriorEditChar
TCustomMaskEdit class 1086
GetPropDisplayString
TOleControl class 2661
GetPropDisplayStrings
TOleControl class 2661
GetProperty
IGetContextProperties interface 2420
TOleControl class 2661
GetPropInfo
TObjectDispatch class 2487
GetProvider
TSQLResolver class 3203
GetPurgeCount
TIBDatabaseInfo class 244

GetReadIdxCount
TIBDatabaseInfo class 244
GetReadOnly
TIBDatabaseInfo class 244
GetReads
TIBDatabaseInfo class 244
GetReadSeqCount
TIBDatabaseInfo class 245
GetRecNo
TIBCustomDataSet class 77
GetRecord
TIBCustomDataSet class 77
GetRecordCount
TIBCustomDataSet class 78
TIBSQL class 526
GetRecords
TCustomProvider class 3109
GetRecordSize
TIBCustomDataSet class 78
TIBXSQLDA class 541
GetRegInterfaceEntry
TInvokableClassRegistry class 854
GetReplyBuffer
ISkeleton interface 2810
GetRequestHeaderInfoForInterface
TInvokableClassRegistry class 855
GetRequiredTags
TBaseInetXPageProducer class 2267
GetResource
TCustomImageList class 668
GetResourceHandle
TCustomRibbonSkin class 4021
GetResponseHeaderInfoForInterface
TInvokableClassRegistry class 855
GetResponseStream
TRIO class 4062
GetResultPointer
TInvContext class 832
GetReturnParamNames
TInvokableClassRegistry class 856
GetRibbon
IRibbon interface 3567
TRibbonApplicationMenuBar class 3947
TRibbonApplicationPopupMenu class 3966
GetRibbonHeight
TCustomRibbon class 3380
GetRibbonMetric
TCustomRibbon class 3380
GetRIO
IRIOAccess interface 4069
GetRowOfNode
TOutlineNode class 2948
GetRowsAffected
TIBSQL class 526
GetRowSetFieldAttributes
IDataSetField interface 2205
TWebDataInput class 2091
GetScriptManager
TBaseInetXPageProducer class 2268
GetScrollBtnClass
TCustomRibbonQuickAccessToolbar class 3449
TPlatformDefaultStyleActionBars class 2978
TRibbonStyleActionBars class 4027
GetSel
TCustomMaskEdit class 1087
GetSelectIndex
TQuerySelectOptions class 1992
TWebSelectOptionsInput class 2132
GetServer
TDispatchConnection class 1293
TOleServer class 2714
TSharedConnection class 1313
GetServerCLSID
TDispatchConnection class 1293
GetServerList
TDispatchConnection class 1294
GetShort
IMarshalInBuffer interface 2799

GetShortIntProp
TOleControl class 2662
GetShowShortCut
TRibbonMenuItem class 3759
TRibbonRichContentMainMenuItem class 3783
GetSingleProp
TOleControl class 2662
GetSize
TStream class 3006
GetSkeleton
ISkeletonObject interface 2811
GetSkinClass
TRibbonLunaStyleActionBars class 4001
TRibbonObsidianStyleActionBars class 4007
TRibbonSilverStyleActionBars class 4013
TRibbonStyleActionBars class 4028
GetSmallintProp
TOleControl class 2662
GetSourceStream
TSOAPAttachment class 899
GetSQL
TIBDataSetUpdateObject class 137
TIBUpdateSQL class 633
GetSQLParams
TIBSQL class 526
GetStateFieldValue
TPacketDataSet class 3191
GetStatusCode
TISAPIResponse class 980
GetStream
TWebRequestFile class 3335
GetString
TCustomListAction class 1022
TCustomStaticListAction class 1031
TCustomVirtualListAction class 1035
GetStringDatabaseInfo
TIBDatabaseInfo class 245
GetStringProp
TOleControl class 2663
GetStrings
TEnumPropDesc class 2633
TMemIniFile class 772
GetStringVariable
TISAPIRequest class 973
TISAPIResponse class 981
GetStub
IStubObject interface 2816
GetStyleName
TPlatformDefaultStyleActionBars class 2978
TRibbonLunaStyleActionBars class 4001
TRibbonObsidianStyleActionBars class 4007
TRibbonSilverStyleActionBars class 4013
GetSubComponents
TApplyUpdatesButton class 1800
TDataSetButton class 1863
TWebControlGroup class 2069
TWebForm class 2108
TWebListInput class 2118
GetSweepInterval
TIBDatabaseInfo class 245
GetTabAt
TCustomRibbon class 3381
GetTableNames
TIBDataBase class 185
GetTabRect
TCustomRibbon class 3381
GetTabWidth
TCustomRibbon class 3381
GetTColorProp
TOleControl class 2663
GetTDateTimeProp
TOleControl class 2664
GetTempDir
IOPConvert interface 2732
GetTemplateFileName
TCustomInetXPageProducer class 2280
GetTemplateStream
TCustomInetXPageProducer class 2280

TPageItemsProducer class 2972
GetText
IMarshalInBuffer interface 2799
IQueryField interface 2218
TQueryRadioGroup class 1980
TQuerySelectOptions class 1992
TQueryText class 2004
TQueryTextArea class 2014
GetTextItem
TCustomOutline class 2852
TOutlineNode class 2948
GetTextLen
TCustomMaskEdit class 1087
GetTFontProp
TOleControl class 2664
GetTOleEnumProp
TOleControl class 2664
GetTopPosition
IRibbonGroupDropdown interface 3918
TRibbonGroupCollapsedControl class 3511
TRibbonGroupDropDown class 3741
GetTPictureProp
TOleControl class 2665
GetTraceFlags
IIBSQLMonitorHook interface 569
GetTransaction
TIBBlobStream class 45
TIBSQL class 527
GetTRHandle
TIBBase class 172
TIBBlobStream class 46
TIBSQL class 527
GetTypeBySchemaNS
TSOAPDomConv class 2772
GetTypeInfo
TAutoDispatch class 2502
TEventDispatch class 2636
TLocalAppServer class 3187
TObjectDispatch class 2487
TOleControl class 2665
TServerEventDispatch class 2722
GetTypeInfoCount
TAutoDispatch class 2502
TEventDispatch class 2637
TLocalAppServer class 3187
TObjectDispatch class 2487
TOleControl class 2665
TServerEventDispatch class 2722
GetUDDIInfo
TInvokableClassRegistry class 856
GetUniqueRelationName
TIBSQL class 527
GetUnsignedChar
IMarshalInBuffer interface 2799
GetUnsignedLong
IMarshalInBuffer interface 2800
GetUnsignedShort
IMarshalInBuffer interface 2800
GetUpdateCount
TIBDatabaseInfo class 245
GetURICount
TRemotableTypeRegistry class 879
GetURIMap
TRemotableTypeRegistry class 880
GetUserNames
TIBDatabaseInfo class 246
GetUseXMLIslands
TBaseInetXPageProducer class 2268
GetValue
TIBStringField class 148
GetValueNames
TRegistry class 3265
GetVariantProp
TOleControl class 2666
GetVarTypeFromXSD
TRemotableTypeRegistry class 881
GetVersion
TIBDatabaseInfo class 246

GetVisibleFields
IDataSetFields interface 2207
TWebForm class 2108
TXMLDisplayGroup class 2181
GetVisibleNode
TCustomOutline class 2852
TOutlineNode class 2949
GetWebDisplayComponent
IWebDisplayReference interface 2228
TWebDisplayReferenceGroup class 2100
GetWidechar
IMarshalInBuffer interface 2800
GetWideStringProp
TOleControl class 2666
GetWideText
IMarshalInBuffer interface 2800
GetWidth
TJPEGImage class 1002
GetWindowContext
TOleControl class 2667
GetWordBoolProp
TOleControl class 2667
GetWordProp
TOleControl class 2667
GetWrites
TIBDatabaseInfo class 246
GetWSDLEncoding
TInvokableClassRegistry class 857
GetXMLBroker
IDataSetComponent interface 2201
TXMLButton class 2154
TXMLData class 2159
TXMLDataParent class 2163
TXMLDataSetParent class 2174
TXMLDisplayGroup class 2181
GetXMLComponent
TQueryButton class 1957
GetXMLData
TBaseInetXPageProducer class 2268
GetXMLDataSet
IDataSetComponent interface 2202
TXMLDisplayGroup class 2182
GetXMLDelta
TBaseInetXPageProducer class 2268
GetXmlDisplayName
TWebDataDisplay class 2079
TXMLDisplayReferenceButton class 2194
GetXMLDisplayName
IXMLDisplay interface 2230
TXMLDisplayGroup class 2182
GetXMLErrors
TBaseInetXPageProducer class 2269
GetXmlRowSetName
TWebDataDisplay class 2080
GetXMLRowSetName
IXMLDisplay interface 2230
TXMLDisplayGroup class 2182
GetXMLUseParent
TXMLButton class 2154
GetXSDInfoForClass
TRemotableTypeRegistry class 881
GetXSQLDA
TIBXSQLDA class 541
GetXSQLVAR
TIBXSQLDA class 541
GetXSQLVARByName
TIBXSQLDA class 541
GlobalAction
TIBValidationService class 465
GotoCurrent
TIBTable class 610
GoToFirstRecordOnExecute
TIBSQL class 532
Grayscale
TJPEGImage class 1008
grMetaData enumeration member 3217
Group
ISharedPropertyGroupManager interface 2433

TQuickAccessToolbarGroupItem class 3477
TQuickAccessToolbarGroups class 3479
TRibbonGroupCollapsedControl class 3518
TRibbonGroupDropDown class 3742
GroupAlign
TCustomRibbonGroup class 3421
TRibbonGroup class 3504
GroupCount
TCustomRibbonPage class 3437
GroupFontColor
TCustomRibbonColorMap class 3404
TRibbonLunaColorMap class 3529
TRibbonObsidianColorMap class 3535
TRibbonSilverColorMap class 3553
GroupID
TIBSecurityService class 449
TUserInfo class 474
GroupIndex
TCustomRibbonGroup class 3422
TMenuItem class 1408
TRibbonGroup class 3504
GroupOffset
TCustomRibbonPage class 3437
Groups
TCustomRibbonPage class 3437
GroupVariant
TCustomRibbonGroup class 3422
GroupWidth
TCustomRibbonGroup class 3422
grReset enumeration member 3217
grXML enumeration member 3217
grXMLUTF8 enumeration member 3217

H

Handle
TCustomImageList class 682
TIBBlobStream class 52
TIBCustomService class 430
TIBCustomSQLMonitor class 563
TIBDataBase class 191
TIBSQL class 532
TIBTransaction class 217
TMenu class 1345
TMenuItem class 1409
TPrinter class 3054
HandleAllocated
TCustomImageList class 669
Handled
TKeyTipMatchItem class 3599
HandleIsShared
TIBDataBase class 191
TIBTransaction class 217
HandleKeyTip
IRibbonKeyTip interface 3920
TCustomRibbonGroup class 3415
TRibbonApplicationMenuButton class 3608
TRibbonBaseButtonControl class 3623
TRibbonComboControl class 3691
TRibbonCustomControl class 3703
TRibbonGroupCollapsedControl class 3511
HandleKeyTips
TCustomRibbonPage class 3432
HandleNeeded
TCustomImageList class 669
HandlesTarget
TCustomListAction class 1022
HandleUpdateError
TCustomResolver class 3131
Has_COMPUTED_BLR
TIBDataBase class 185
TIBSchema class 202
Has_DEFAULT_VALUE
TIBDataBase class 186
TIBSchema class 202
HasApplicationMenu
TCustomRibbon class 3381
HasAsParent
TOutlineNode class 2949

HasCurValues
TPacketDataSet class 3192
HasErrors
TUpdateTree class 3211
HasFailed
TServerItem class 2474
HasGallery
TCustomRibbonGroup class 3423
Hash
IStub interface 2813
HashOf
TStringHash class 785
HasItems
TOutlineNode class 2953
HasMergeConflicts
TPacketDataSet class 3192
HasParent
TMenuItem class 1381
HasQuickAccessToolbar
TCustomRibbon class 3382
HasRegInterfaceImpl
TInvokableClassRegistry class 858
HasStatusField
IDataSetFields interface 2207
TXMLDisplayGroup class 2182
HasSubKeys
TRegistry class 3266
HeaderPages enumeration member 482
Headers
THeaderList class 828
TSOAPAttachment class 906
HeadingAttributes
TCustomDataGrid class 1817
TDataGrid class 1854
Height
TCustomImageList class 682
TRibbon class 3493
TRibbonApplicationMenuBar class 3954
TRibbonGroup class 3505
TRibbonPage class 3541
HelpContext
TMenuItem class 1410
TPopupMenu class 1438
TStaticListAction class 1058
TVirtualListAction class 1069
HelpFile
EOleException class 2499
HideControls
TCustomRibbonGroup class 3416
HideKeyTip
IRibbonKeyTip interface 3920
TCustomRibbonGroup class 3416
TRibbonApplicationMenuButton class 3608
TRibbonBaseButtonControl class 3623
TRibbonComboControl class 3692
TRibbonCustomControl class 3704
TRibbonGroupCollapsedControl class 3512
HideKeyTips
TCustomRibbonPage class 3432
HideRibbon
TCustomRibbon class 3396
HideSelection
TRibbonComboBox class 3633
HideShadow
TRibbonQATCustomizeMenu class 3969
HideTabs
TCustomRibbon class 3396
TRibbon class 3495
HighlightColor
TRibbonLunaColorMap class 3529
TRibbonObsidianColorMap class 3535
TRibbonSilverColorMap class 3554
Hint
TMenuItem class 1410
TOptionItem class 3937
TStaticListAction class 1058
TVirtualListAction class 1069
hmtAll enumeration member 934

hmtRequest enumeration member 934
hmtResponse enumeration member 934
HorzMargin
TRibbonApplicationMenuBar class 3955
HostSite
TLimboTransactionInfo class 471
HotTabIndex
TCustomRibbon class 3396
HTMLControlName
IHTMLField interface 2211
HTMLDoc
TCustomInetXPageProducer class 2281
TInetXPageProducer class 2288
TPageItemsProducer class 2972
HTMLFile
TCustomInetXPageProducer class 2282
TInetXPageProducer class 2289
TPageItemsProducer class 2973
HTMLFormName
IHTMLForm interface 2213
HTMLFormVarName
IHTMLForm interface 2213
HttpExtensionProc
TISAPIApplication class 962

I

IB 1
IB namespace 1
Classes 3
Constants 21
Functions 7
Structs, Records, Enums 11
Types 17
Variables 20
IB.CheckStatusVector 7
IB.CheckStatusVector function 7
IB.EIBClientError 3
IB.EIBError 3
IB.EIBError.Create 4
IB.EIBError.IBErrorCode 5
IB.EIBError.SQLCode 5
IB.EIBInterBaseError 5
IB.EIBInterBaseRoleError 6
IB.GetIBDataBaseErrorMessages 7
IB.GetIBDataBaseErrorMessages function 7
IB.IBAlloc 8
IB.IBAlloc function 8
IB.IBBigLocalBufferLength 21
IB.IBBigLocalBufferLength constant 21
IB.IBCS 20
IB.IBCS variable 20
IB.IBDataBaseError 8
IB.IBDataBaseError function 8
IB.IBError 8
IB.IBError function 8
IB.IBErrorMessages 21
IB.IBErrorMessages constant 21
IB.IBHugeLocalBufferLength 22
IB.IBHugeLocalBufferLength constant 22
IB.IBLocalBufferLength 23
IB.IBLocalBufferLength constant 23
IB.IBPalette1 23
IB.IBPalette1 constant 23
IB.IBPalette2 23
IB.IBPalette2 constant 23
IB.PStatusVector 17
IB.PStatusVector type 17
IB.SetIBDataBaseErrorMessages 9
IB.SetIBDataBaseErrorMessages function 9
IB.StatusVector 9
IB.StatusVector function 9
IB.StatusVectorArray 9
IB.StatusVectorArray function 9
IB.StatusVectorAsText 10
IB.StatusVectorAsText function 10
IB.TIBClientError 11
IB.TIBClientError enumeration 11
IB.TIBDataBaseErrorMessage 15

IB.TIBDataBaseErrorMessage enumeration 15
IB.TIBDataBaseErrorMessages 17
IB.TIBDataBaseErrorMessages type 17
IB.TStatusVector 18
IB.TStatusVector type 18
IB.TTraceFlag 15
IB.TTraceFlag enumeration 15
IB.TTraceFlags 18
IB.TTraceFlags type 18
IBBatchMove 25
IBBatchMove namespace 25
Classes 26
Structs, Records, Enums 38
IBBatchMove.TBatchMode 38
IBBatchMove.TBatchMode enumeration 38
IBBatchMove.TIBBatchMove 26
IBBatchMove.TIBBatchMove.AbortOnKeyViol 30
IBBatchMove.TIBBatchMove.AbortOnProblem 30
IBBatchMove.TIBBatchMove.ChangedCount 31
IBBatchMove.TIBBatchMove.ChangedTableName 31
IBBatchMove.TIBBatchMove.CommitCount 31
IBBatchMove.TIBBatchMove.Create 27
IBBatchMove.TIBBatchMove.Destination 32
IBBatchMove.TIBBatchMove.Destroy 28
IBBatchMove.TIBBatchMove.Execute 28
IBBatchMove.TIBBatchMove.KeyViolCount 32
IBBatchMove.TIBBatchMove.KeyViolTableName 33
IBBatchMove.TIBBatchMove.Mappings 33
IBBatchMove.TIBBatchMove.Mode 34
IBBatchMove.TIBBatchMove.MovedCount 34
IBBatchMove.TIBBatchMove.Notification 29
IBBatchMove.TIBBatchMove.ProblemCount 35
IBBatchMove.TIBBatchMove.ProblemTableName 35
IBBatchMove.TIBBatchMove.RecordCount 36
IBBatchMove.TIBBatchMove.Source 36
IBBatchMove.TIBBatchMove.Transliterate 37
IBBlob 39
IBBlob namespace 39
Classes 40
Constants 56
Functions 54
IBBlob.DefaultBlobSegmentSize 56
IBBlob.DefaultBlobSegmentSize constant 56
IBBlob.GetBlobInfo 54
IBBlob.GetBlobInfo function 54
IBBlob.ReadBlob 54
IBBlob.ReadBlob function 54
IBBlob.TIBBlobStream 40
IBBlob.TIBBlobStream.BlobID 50
IBBlob.TIBBlobStream.BlobMaxSegmentSize 51
IBBlob.TIBBlobStream.BlobNumSegments 51
IBBlob.TIBBlobStream.BlobSize 51
IBBlob.TIBBlobStream.BlobType 51
IBBlob.TIBBlobStream.Call 43
IBBlob.TIBBlobStream.Cancel 43
IBBlob.TIBBlobStream.CheckReadable 43
IBBlob.TIBBlobStream.CheckWritable 44
IBBlob.TIBBlobStream.CloseBlob 44
IBBlob.TIBBlobStream.Create 41
IBBlob.TIBBlobStream.CreateBlob 44
IBBlob.TIBBlobStream.Database 52
IBBlob.TIBBlobStream.DBHandle 52
IBBlob.TIBBlobStream.Destroy 42
IBBlob.TIBBlobStream.EnsureBlobInitialized 44
IBBlob.TIBBlobStream.Finalize 45
IBBlob.TIBBlobStream.GetBlobInfo 45
IBBlob.TIBBlobStream.GetDatabase 45
IBBlob.TIBBlobStream.GetDBHandle 45
IBBlob.TIBBlobStream.GetTransaction 45
IBBlob.TIBBlobStream.GetTRHandle 46
IBBlob.TIBBlobStream.Handle 52
IBBlob.TIBBlobStream.LoadFromFile 46
IBBlob.TIBBlobStream.LoadFromStream 46
IBBlob.TIBBlobStream.Mode 52
IBBlob.TIBBlobStream.Modified 53
IBBlob.TIBBlobStream.OpenBlob 46
IBBlob.TIBBlobStream.Read 47
IBBlob.TIBBlobStream.SaveToFile 47

IBBlob.TIBBlobStream.SaveToStream 47
IBBlob.TIBBlobStream.Seek 48
IBBlob.TIBBlobStream.SetBlobID 48
IBBlob.TIBBlobStream.SetDatabase 48
IBBlob.TIBBlobStream.SetMode 49
IBBlob.TIBBlobStream.SetSize 49
IBBlob.TIBBlobStream.SetTransaction 49
IBBlob.TIBBlobStream.Transaction 53
IBBlob.TIBBlobStream.TRHandle 53
IBBlob.TIBBlobStream.Truncate 49
IBBlob.TIBBlobStream.Write 50
IBBlob.WriteBlob 55
IBBlob.WriteBlob function 55
IBCustomDataSet 57
IBCustomDataSet namespace 57
Classes 59
Constants 160
Structs, Records, Enums 149
Types 154
Variables 159
IBCustomDataSet.BufferCacheSize 160
IBCustomDataSet.BufferCacheSize constant 160
IBCustomDataSet.CreateProviderProc 159
IBCustomDataSet.CreateProviderProc variable 159
IBCustomDataSet.DefaultFieldClasses 160
IBCustomDataSet.DefaultFieldClasses constant 160
IBCustomDataSet.PBlobDataArray 154
IBCustomDataSet.PBlobDataArray type 154
IBCustomDataSet.PDateTime 154
IBCustomDataSet.PDateTime type 154
IBCustomDataSet.PFieldData 155
IBCustomDataSet.PFieldData type 155
IBCustomDataSet.PIBDBKey 155
IBCustomDataSet.PIBDBKey type 155
IBCustomDataSet.PRecordData 155
IBCustomDataSet.PRecordData type 155
IBCustomDataSet.TBlobDataArray 156
IBCustomDataSet.TBlobDataArray type 156
IBCustomDataSet.TCachedUpdateStatus 149
IBCustomDataSet.TCachedUpdateStatus enumeration 149
IBCustomDataSet.TFieldData 150
IBCustomDataSet.TFieldData record 150
IBCustomDataSet.TIBBCDField 59
IBCustomDataSet.TIBBCDField.CheckTypeSize 60
IBCustomDataSet.TIBBCDField.Create 60
IBCustomDataSet.TIBBCDField.GetAsCurrency 61
IBCustomDataSet.TIBBCDField.GetAsString 61
IBCustomDataSet.TIBBCDField.GetAsVariant 61
IBCustomDataSet.TIBBCDField.GetDataSize 61
IBCustomDataSet.TIBBCDField.Size 62
IBCustomDataSet.TIBCustomDataSet 62
IBCustomDataSet.TIBCustomDataSet.ActivateConnection 69
IBCustomDataSet.TIBCustomDataSet.ActivateTransaction 69
IBCustomDataSet.TIBCustomDataSet.AfterCancel 94
IBCustomDataSet.TIBCustomDataSet.AfterClose 94
IBCustomDataSet.TIBCustomDataSet.AfterDatabaseDisconnect 109
IBCustomDataSet.TIBCustomDataSet.AfterDelete 94
IBCustomDataSet.TIBCustomDataSet.AfterEdit 94
IBCustomDataSet.TIBCustomDataSet.AfterInsert 95
IBCustomDataSet.TIBCustomDataSet.AfterOpen 95
IBCustomDataSet.TIBCustomDataSet.AfterPost 95
IBCustomDataSet.TIBCustomDataSet.AfterRefresh 95
IBCustomDataSet.TIBCustomDataSet.AfterScroll 95
IBCustomDataSet.TIBCustomDataSet.AfterTransactionEnd 109
IBCustomDataSet.TIBCustomDataSet.AllocRecordBuffer 69
IBCustomDataSet.TIBCustomDataSet.ApplyUpdates 69
IBCustomDataSet.TIBCustomDataSet.AutoCalcFields 96
IBCustomDataSet.TIBCustomDataSet.BeforeCancel 96
IBCustomDataSet.TIBCustomDataSet.BeforeClose 96
IBCustomDataSet.TIBCustomDataSet.BeforeDatabaseDisconnect 109
IBCustomDataSet.TIBCustomDataSet.BeforeDelete 96
IBCustomDataSet.TIBCustomDataSet.BeforeEdit 97
IBCustomDataSet.TIBCustomDataSet.BeforeInsert 97
IBCustomDataSet.TIBCustomDataSet.BeforeOpen 97
IBCustomDataSet.TIBCustomDataSet.BeforePost 97
IBCustomDataSet.TIBCustomDataSet.BeforeRefresh 97

IBCustomDataSet.TIBCustomDataSet.BeforeScroll 98
IBCustomDataSet.TIBCustomDataSet.BeforeTransactionEnd 109
IBCustomDataSet.TIBCustomDataSet.BookmarkValid 70
IBCustomDataSet.TIBCustomDataSet.BufferChunks 98
IBCustomDataSet.TIBCustomDataSet.CachedUpdates 98
IBCustomDataSet.TIBCustomDataSet.CachedUpdateStatus 70
IBCustomDataSet.TIBCustomDataSet.CancelUpdates 70
IBCustomDataSet.TIBCustomDataSet.CheckDatasetClosed 71
IBCustomDataSet.TIBCustomDataSet.CheckDatasetOpen 71
IBCustomDataSet.TIBCustomDataSet.CheckNotUniDirectional 71
IBCustomDataSet.TIBCustomDataSet.ClearCalcFields 71
IBCustomDataSet.TIBCustomDataSet.CompareBookmarks 71
IBCustomDataSet.TIBCustomDataSet.ConstraintsStored 72
IBCustomDataSet.TIBCustomDataSet.Create 66
IBCustomDataSet.TIBCustomDataSet.CreateBlobStream 72
IBCustomDataSet.TIBCustomDataSet.CreateFields 72
IBCustomDataSet.TIBCustomDataSet.Current 73
IBCustomDataSet.TIBCustomDataSet.Database 99
IBCustomDataSet.TIBCustomDataSet.DatabaseFree 110
IBCustomDataSet.TIBCustomDataSet.DBHandle 99
IBCustomDataSet.TIBCustomDataSet.DeactivateTransaction 73
IBCustomDataSet.TIBCustomDataSet.DeleteSQL 99
IBCustomDataSet.TIBCustomDataSet.Destroy 66
IBCustomDataSet.TIBCustomDataSet.Disconnect 73
IBCustomDataSet.TIBCustomDataSet.DoBeforeDelete 73
IBCustomDataSet.TIBCustomDataSet.DoBeforeEdit 73
IBCustomDataSet.TIBCustomDataSet.DoBeforeInsert 74
IBCustomDataSet.TIBCustomDataSet.DoOnNewRecord 74
IBCustomDataSet.TIBCustomDataSet.FetchAll 74
IBCustomDataSet.TIBCustomDataSet.ForcedRefresh 99
IBCustomDataSet.TIBCustomDataSet.FreeRecordBuffer 75
IBCustomDataSet.TIBCustomDataSet.GeneratorField 100
IBCustomDataSet.TIBCustomDataSet.GetActiveBuf 75
IBCustomDataSet.TIBCustomDataSet.GetBookmarkData 75
IBCustomDataSet.TIBCustomDataSet.GetBookmarkFlag 75
IBCustomDataSet.TIBCustomDataSet.GetCanModify 75
IBCustomDataSet.TIBCustomDataSet.GetCurrentRecord 76
IBCustomDataSet.TIBCustomDataSet.GetDataSource 76
IBCustomDataSet.TIBCustomDataSet.GetFieldClass 76
IBCustomDataSet.TIBCustomDataSet.GetFieldData 77
IBCustomDataSet.TIBCustomDataSet.GetRecNo 77
IBCustomDataSet.TIBCustomDataSet.GetRecord 77
IBCustomDataSet.TIBCustomDataSet.GetRecordCount 78
IBCustomDataSet.TIBCustomDataSet.GetRecordSize 78
IBCustomDataSet.TIBCustomDataSet.InitRecord 78
IBCustomDataSet.TIBCustomDataSet.InsertSQL 100
IBCustomDataSet.TIBCustomDataSet.InternalAddRecord 78
IBCustomDataSet.TIBCustomDataSet.InternalBatchInput 79
IBCustomDataSet.TIBCustomDataSet.InternalBatchOutput 79
IBCustomDataSet.TIBCustomDataSet.InternalCancel 79
IBCustomDataSet.TIBCustomDataSet.InternalClose 79
IBCustomDataSet.TIBCustomDataSet.InternalDelete 79
IBCustomDataSet.TIBCustomDataSet.InternalExecQuery 80
IBCustomDataSet.TIBCustomDataSet.InternalFirst 80
IBCustomDataSet.TIBCustomDataSet.InternalGotoBookmark 80
IBCustomDataSet.TIBCustomDataSet.InternalHandleException 80
IBCustomDataSet.TIBCustomDataSet.InternalInitFieldDefs 81
IBCustomDataSet.TIBCustomDataSet.InternalInitRecord 81
IBCustomDataSet.TIBCustomDataSet.InternalInsert 81
IBCustomDataSet.TIBCustomDataSet.InternalLast 81
IBCustomDataSet.TIBCustomDataSet.InternalOpen 81
IBCustomDataSet.TIBCustomDataSet.InternalPost 82
IBCustomDataSet.TIBCustomDataSet.InternalPrepare 82
IBCustomDataSet.TIBCustomDataSet.InternalPrepared 100
IBCustomDataSet.TIBCustomDataSet.InternalRefresh 82
IBCustomDataSet.TIBCustomDataSet.InternalRefreshRow 82
IBCustomDataSet.TIBCustomDataSet.InternalSetParamsFromCursor 83
IBCustomDataSet.TIBCustomDataSet.InternalSetToRecord 83
IBCustomDataSet.TIBCustomDataSet.InternalUnPrepare 83
IBCustomDataSet.TIBCustomDataSet.IsCursorOpen 83
IBCustomDataSet.TIBCustomDataSet.IsSequenced 83
IBCustomDataSet.TIBCustomDataSet.LiveMode 101
IBCustomDataSet.TIBCustomDataSet.Loaded 84
IBCustomDataSet.TIBCustomDataSet.Locate 84
IBCustomDataSet.TIBCustomDataSet.LocateNext 85

IBCustomDataSet.TIBCustomDataSet.Lookup 85
IBCustomDataSet.TIBCustomDataSet.ModifySQL 101
IBCustomDataSet.TIBCustomDataSet.ObjectView 101
IBCustomDataSet.TIBCustomDataSet.OnCalcFields 101
IBCustomDataSet.TIBCustomDataSet.OnDeleteError 102
IBCustomDataSet.TIBCustomDataSet.OnEditError 102
IBCustomDataSet.TIBCustomDataSet.OnNewRecord 102
IBCustomDataSet.TIBCustomDataSet.OnPostError 102
IBCustomDataSet.TIBCustomDataSet.OnUpdateError 110
IBCustomDataSet.TIBCustomDataSet.OnUpdateRecord 111
IBCustomDataSet.TIBCustomDataSet.ParamCheck 102
IBCustomDataSet.TIBCustomDataSet.Params 103
IBCustomDataSet.TIBCustomDataSet.Plan 103
IBCustomDataSet.TIBCustomDataSet.Post 86
IBCustomDataSet.TIBCustomDataSet.PSEndTransaction 86
IBCustomDataSet.TIBCustomDataSet.PSExecuteStatement 86
IBCustomDataSet.TIBCustomDataSet.PSGetQuoteChar 87
IBCustomDataSet.TIBCustomDataSet.PSGetTableName 87
IBCustomDataSet.TIBCustomDataSet.PSGetUpdateException 87
IBCustomDataSet.TIBCustomDataSet.PSInTransaction 87
IBCustomDataSet.TIBCustomDataSet.PSIsSQLBased 88
IBCustomDataSet.TIBCustomDataSet.PSIsSQLSupported 88
IBCustomDataSet.TIBCustomDataSet.PSReset 88
IBCustomDataSet.TIBCustomDataSet.PSStartTransaction 88
IBCustomDataSet.TIBCustomDataSet.PSUpdateRecord 88
IBCustomDataSet.TIBCustomDataSet.QDelete 103
IBCustomDataSet.TIBCustomDataSet.QInsert 104
IBCustomDataSet.TIBCustomDataSet.QModify 104
IBCustomDataSet.TIBCustomDataSet.QRefresh 104
IBCustomDataSet.TIBCustomDataSet.QSelect 104
IBCustomDataSet.TIBCustomDataSet.RecordModified 89
IBCustomDataSet.TIBCustomDataSet.RefreshSQL 105
IBCustomDataSet.TIBCustomDataSet.ReQuery 89
IBCustomDataSet.TIBCustomDataSet.RevertRecord 89
IBCustomDataSet.TIBCustomDataSet.RowsAffected 105
IBCustomDataSet.TIBCustomDataSet.SelectSQL 105
IBCustomDataSet.TIBCustomDataSet.SelectStmtHandle 105
IBCustomDataSet.TIBCustomDataSet.SetActive 89
IBCustomDataSet.TIBCustomDataSet.SetBookmarkData 90
IBCustomDataSet.TIBCustomDataSet.SetBookmarkFlag 90
IBCustomDataSet.TIBCustomDataSet.SetCachedUpdates 90
IBCustomDataSet.TIBCustomDataSet.SetDataSource 90
IBCustomDataSet.TIBCustomDataSet.SetFieldData 91
IBCustomDataSet.TIBCustomDataSet.SetRecNo 91
IBCustomDataSet.TIBCustomDataSet.SQLParams 106
IBCustomDataSet.TIBCustomDataSet.SQLType 91
IBCustomDataSet.TIBCustomDataSet.StatementType 106
IBCustomDataSet.TIBCustomDataSet.Transaction 106
IBCustomDataSet.TIBCustomDataSet.TransactionFree 111
IBCustomDataSet.TIBCustomDataSet.TRHandle 106
IBCustomDataSet.TIBCustomDataSet.Undelete 92
IBCustomDataSet.TIBCustomDataSet.UniDirectional 107
IBCustomDataSet.TIBCustomDataSet.UpdateMode 107
IBCustomDataSet.TIBCustomDataSet.UpdateObject 107
IBCustomDataSet.TIBCustomDataSet.UpdateRecordTypes 108
IBCustomDataSet.TIBCustomDataSet.UpdatesPending 108
IBCustomDataSet.TIBCustomDataSet.UpdateStatus 92
IBCustomDataSet.TIBDataLink 111
IBCustomDataSet.TIBDataLink.ActiveChanged 113
IBCustomDataSet.TIBDataLink.CheckBrowseMode 113
IBCustomDataSet.TIBDataLink.Create 112
IBCustomDataSet.TIBDataLink.Destroy 112
IBCustomDataSet.TIBDataLink.GetDetailDataSet 113
IBCustomDataSet.TIBDataLink.RecordChanged 114
IBCustomDataSet.TIBDataSet 114
IBCustomDataSet.TIBDataSet.Active 122
IBCustomDataSet.TIBDataSet.AfterCancel 122
IBCustomDataSet.TIBDataSet.AfterClose 122
IBCustomDataSet.TIBDataSet.AfterDatabaseDisconnect 133
IBCustomDataSet.TIBDataSet.AfterDelete 123
IBCustomDataSet.TIBDataSet.AfterEdit 123
IBCustomDataSet.TIBDataSet.AfterInsert 123
IBCustomDataSet.TIBDataSet.AfterOpen 123
IBCustomDataSet.TIBDataSet.AfterPost 123
IBCustomDataSet.TIBDataSet.AfterScroll 124
IBCustomDataSet.TIBDataSet.AfterTransactionEnd 133
IBCustomDataSet.TIBDataSet.AutoCalcFields 124
IBCustomDataSet.TIBDataSet.BatchInput 119

IBCustomDataSet.TIBDataSet.BatchOutput 119
IBCustomDataSet.TIBDataSet.BeforeCancel 124
IBCustomDataSet.TIBDataSet.BeforeClose 124
IBCustomDataSet.TIBDataSet.BeforeDatabaseDisconnect 133
IBCustomDataSet.TIBDataSet.BeforeDelete 125
IBCustomDataSet.TIBDataSet.BeforeEdit 125
IBCustomDataSet.TIBDataSet.BeforeInsert 125
IBCustomDataSet.TIBDataSet.BeforeOpen 125
IBCustomDataSet.TIBDataSet.BeforePost 125
IBCustomDataSet.TIBDataSet.BeforeScroll 126
IBCustomDataSet.TIBDataSet.BeforeTransactionEnd 134
IBCustomDataSet.TIBDataSet.BufferChunks 126
IBCustomDataSet.TIBDataSet.CachedUpdates 126
IBCustomDataSet.TIBDataSet.DatabaseFree 134
IBCustomDataSet.TIBDataSet.DataSource 127
IBCustomDataSet.TIBDataSet.DeleteSQL 127
IBCustomDataSet.TIBDataSet.ExecSQL 119
IBCustomDataSet.TIBDataSet.Filtered 127
IBCustomDataSet.TIBDataSet.GeneratorField 127
IBCustomDataSet.TIBDataSet.InsertSQL 128
IBCustomDataSet.TIBDataSet.InternalOpen 119
IBCustomDataSet.TIBDataSet.LiveMode 128
IBCustomDataSet.TIBDataSet.ModifySQL 128
IBCustomDataSet.TIBDataSet.OnCalcFields 129
IBCustomDataSet.TIBDataSet.OnDeleteError 129
IBCustomDataSet.TIBDataSet.OnEditError 129
IBCustomDataSet.TIBDataSet.OnFilterRecord 129
IBCustomDataSet.TIBDataSet.OnNewRecord 129
IBCustomDataSet.TIBDataSet.OnPostError 130
IBCustomDataSet.TIBDataSet.ParamByName 120
IBCustomDataSet.TIBDataSet.ParamCheck 130
IBCustomDataSet.TIBDataSet.Params 130
IBCustomDataSet.TIBDataSet.Prepare 120
IBCustomDataSet.TIBDataSet.Prepared 131
IBCustomDataSet.TIBDataSet.PSSetCommandText 120
IBCustomDataSet.TIBDataSet.RefreshSQL 131
IBCustomDataSet.TIBDataSet.SelectSQL 131
IBCustomDataSet.TIBDataSet.SelectStmtHandle 131
IBCustomDataSet.TIBDataSet.SetFiltered 120
IBCustomDataSet.TIBDataSet.StatementType 132
IBCustomDataSet.TIBDataSet.TransactionFree 134
IBCustomDataSet.TIBDataSet.UniDirectional 132
IBCustomDataSet.TIBDataSet.UnPrepare 121
IBCustomDataSet.TIBDataSet.UpdateObject 132
IBCustomDataSet.TIBDataSetUpdateObject 134
IBCustomDataSet.TIBDataSetUpdateObject.Apply 136
IBCustomDataSet.TIBDataSetUpdateObject.Create 135
IBCustomDataSet.TIBDataSetUpdateObject.DataSet 137
IBCustomDataSet.TIBDataSetUpdateObject.Destroy 136
IBCustomDataSet.TIBDataSetUpdateObject.GetDataSet 136
IBCustomDataSet.TIBDataSetUpdateObject.GetSQL 137
IBCustomDataSet.TIBDataSetUpdateObject.RefreshSQL 137
IBCustomDataSet.TIBDataSetUpdateObject.SetDataSet 137
IBCustomDataSet.TIBDBKey 150
IBCustomDataSet.TIBDBKey record 150
IBCustomDataSet.TIBDSBlobStream 138
IBCustomDataSet.TIBDSBlobStream.Create 139
IBCustomDataSet.TIBDSBlobStream.Destroy 139
IBCustomDataSet.TIBDSBlobStream.FBlobStream 139
IBCustomDataSet.TIBDSBlobStream.FField 140
IBCustomDataSet.TIBDSBlobStream.FModified 140
IBCustomDataSet.TIBDSBlobStream.Read 140
IBCustomDataSet.TIBDSBlobStream.Seek 141
IBCustomDataSet.TIBDSBlobStream.SetSize 141
IBCustomDataSet.TIBDSBlobStream.Write 141
IBCustomDataSet.TIBGeneratorApplyEvent 151
IBCustomDataSet.TIBGeneratorApplyEvent enumeration 151
IBCustomDataSet.TIBGeneratorField 142
IBCustomDataSet.TIBGeneratorField.Apply 143
IBCustomDataSet.TIBGeneratorField.ApplyEvent 145
IBCustomDataSet.TIBGeneratorField.Assign 144
IBCustomDataSet.TIBGeneratorField.Create 143
IBCustomDataSet.TIBGeneratorField.Field 145
IBCustomDataSet.TIBGeneratorField.Generator 145
IBCustomDataSet.TIBGeneratorField.IncrementBy 146
IBCustomDataSet.TIBGeneratorField.ValueName 144
IBCustomDataSet.TIBStringField 146
IBCustomDataSet.TIBStringField.CheckTypeSize 147

IBCustomDataSet.TIBStringField.Create 147
IBCustomDataSet.TIBStringField.GetAsString 148
IBCustomDataSet.TIBStringField.GetAsVariant 148
IBCustomDataSet.TIBStringField.GetValue 148
IBCustomDataSet.TIBStringField.SetAsString 148
IBCustomDataSet.TIBUpdateAction 151
IBCustomDataSet.TIBUpdateAction enumeration 151
IBCustomDataSet.TIBUpdateErrorEvent 156
IBCustomDataSet.TIBUpdateErrorEvent type 156
IBCustomDataSet.TIBUpdateRecordEvent 157
IBCustomDataSet.TIBUpdateRecordEvent type 157
IBCustomDataSet.TIBUpdateRecordTypes 157
IBCustomDataSet.TIBUpdateRecordTypes type 157
IBCustomDataSet.TLiveMode 152
IBCustomDataSet.TLiveMode enumeration 152
IBCustomDataSet.TLiveModes 157
IBCustomDataSet.TLiveModes type 157
IBCustomDataSet.TRecordData 153
IBCustomDataSet.TRecordData record 153
IBCustomDataSet.UniCache 161
IBCustomDataSet.UniCache constant 161
IBDatabase 163
IBDatabase namespace 163
- Classes 165
- Constants 229
- Functions 224
- Interfaces 221
- Structs, Records, Enums 225
- Types 227

IBDatabase.DPBConstantNames 229
IBDatabase.DPBConstantNames constant 229
IBDatabase.DPBPrefix 230
IBDatabase.DPBPrefix constant 230
IBDatabase.GenerateDPB 224
IBDatabase.GenerateDPB function 224
IBDatabase.GenerateTPB 224
IBDatabase.GenerateTPB function 224
IBDatabase.IIBEventNotifier 221
IBDatabase.IIBEventNotifier.GetAutoRegister 222
IBDatabase.IIBEventNotifier.RegisterEvents 222
IBDatabase.IIBEventNotifier.UnRegisterEvents 222
IBDatabase.TAutoStopAction 225
IBDatabase.TAutoStopAction enumeration 225
IBDatabase.TIBBase 165
IBDatabase.TIBBase.AfterDatabaseDisconnect 175
IBDatabase.TIBBase.AfterTransactionEnd 175
IBDatabase.TIBBase.BeforeDatabaseDisconnect 175
IBDatabase.TIBBase.BeforeTransactionEnd 175
IBDatabase.TIBBase.CheckDatabase 170
IBDatabase.TIBBase.CheckTransaction 170
IBDatabase.TIBBase.Create 166
IBDatabase.TIBBase.Database 173
IBDatabase.TIBBase.DBHandle 173
IBDatabase.TIBBase.Destroy 167
IBDatabase.TIBBase.DoAfterDatabaseDisconnect 171
IBDatabase.TIBBase.DoAfterTransactionEnd 171
IBDatabase.TIBBase.DoBeforeDatabaseDisconnect 171
IBDatabase.TIBBase.DoBeforeTransactionEnd 171
IBDatabase.TIBBase.DoDatabaseFree 171
IBDatabase.TIBBase.DoTransactionFree 172
IBDatabase.TIBBase.FAfterDatabaseDisconnect 167
IBDatabase.TIBBase.FAfterTransactionEnd 167
IBDatabase.TIBBase.FBeforeDatabaseDisconnect 168
IBDatabase.TIBBase.FBeforeTransactionEnd 168
IBDatabase.TIBBase.FDatabase 168
IBDatabase.TIBBase.FIndexInDatabase 168
IBDatabase.TIBBase.FIndexInTransaction 169
IBDatabase.TIBBase.FOnDatabaseFree 169
IBDatabase.TIBBase.FOnTransactionFree 169
IBDatabase.TIBBase.FOwner 169
IBDatabase.TIBBase.FTransaction 169
IBDatabase.TIBBase.GetDBHandle 172
IBDatabase.TIBBase.GetTRHandle 172
IBDatabase.TIBBase.OnDatabaseFree 176
IBDatabase.TIBBase.OnTransactionFree 176
IBDatabase.TIBBase.Owner 174
IBDatabase.TIBBase.SetDatabase 172
IBDatabase.TIBBase.SetTransaction 173

IBDatabase.TIBBase.Transaction 174
IBDatabase.TIBBase.TRHandle 174
IBDatabase.TIBDataBase 176
IBDatabase.TIBDataBase.AddEventNotifier 180
IBDatabase.TIBDataBase.AddTransaction 180
IBDatabase.TIBDataBase.AfterConnect 198
IBDatabase.TIBDataBase.AfterDisconnect 199
IBDatabase.TIBDataBase.AllowStreamedConnected 189
IBDatabase.TIBDataBase.ApplyUpdates 180
IBDatabase.TIBDataBase.BeforeConnect 199
IBDatabase.TIBDataBase.BeforeDisconnect 199
IBDatabase.TIBDataBase.Call 181
IBDatabase.TIBDataBase.CheckActive 181
IBDatabase.TIBDataBase.CheckDatabaseName 181
IBDatabase.TIBDataBase.CheckInactive 181
IBDatabase.TIBDataBase.CloseDataSets 182
IBDatabase.TIBDataBase.Connected 189
IBDatabase.TIBDataBase.Create 178
IBDatabase.TIBDataBase.CreateDatabase 182
IBDatabase.TIBDataBase.DatabaseName 190
IBDatabase.TIBDataBase.DBParamByDPB 190
IBDatabase.TIBDataBase.DBSQLDialect 190
IBDatabase.TIBDataBase.DefaultTransaction 191
IBDatabase.TIBDataBase.Destroy 178
IBDatabase.TIBDataBase.DoConnect 182
IBDatabase.TIBDataBase.DoDisconnect 183
IBDatabase.TIBDataBase.DropDatabase 183
IBDatabase.TIBDataBase.FindDefaultTransaction 183
IBDatabase.TIBDataBase.FindTransaction 184
IBDatabase.TIBDataBase.FlushSchema 184
IBDatabase.TIBDataBase.ForceClose 184
IBDatabase.TIBDataBase.GetConnected 184
IBDatabase.TIBDataBase.GetFieldNames 185
IBDatabase.TIBDataBase.GetTableNames 185
IBDatabase.TIBDataBase.Handle 191
IBDatabase.TIBDataBase.HandleIsShared 191
IBDatabase.TIBDataBase.Has_COMPUTED_BLR 185
IBDatabase.TIBDataBase.Has_DEFAULT_VALUE 186
IBDatabase.TIBDataBase.IdleTimer 191
IBDatabase.TIBDataBase.IndexOfDBConst 186
IBDatabase.TIBDataBase.InternalTransaction 192
IBDatabase.TIBDataBase.IsReadOnly 192
IBDatabase.TIBDataBase.Loaded 186
IBDatabase.TIBDataBase.LoginPrompt 192
IBDatabase.TIBDataBase.Notification 187
IBDatabase.TIBDataBase.OnDialectDowngradeWarning 200
IBDatabase.TIBDataBase.OnIdleTimer 200
IBDatabase.TIBDataBase.OnLogin 201
IBDatabase.TIBDataBase.Params 196
IBDatabase.TIBDataBase.RemoveEventNotifier 187
IBDatabase.TIBDataBase.RemoveTransaction 187
IBDatabase.TIBDataBase.RemoveTransactions 188
IBDatabase.TIBDataBase.SetHandle 188
IBDatabase.TIBDataBase.SQLDialect 196
IBDatabase.TIBDataBase.SQLObjectCount 197
IBDatabase.TIBDataBase.SQLObjects 197
IBDatabase.TIBDataBase.TestConnected 188
IBDatabase.TIBDataBase.TraceFlags 197
IBDatabase.TIBDataBase.TransactionCount 198
IBDatabase.TIBDataBase.Transactions 198
IBDatabase.TIBDatabaseLoginEvent 227
IBDatabase.TIBDatabaseLoginEvent type 227
IBDatabase.TIBFileName 227
IBDatabase.TIBFileName type 227
IBDatabase.TIBSchema 201
IBDatabase.TIBSchema.FreeNodes 202
IBDatabase.TIBSchema.Has_COMPUTED_BLR 202
IBDatabase.TIBSchema.Has_DEFAULT_VALUE 202
IBDatabase.TIBTimer 202
IBDatabase.TIBTimer.Create 203
IBDatabase.TIBTimer.Destroy 204
IBDatabase.TIBTimer.Enabled 205
IBDatabase.TIBTimer.Interval 206
IBDatabase.TIBTimer.OnTimer 206
IBDatabase.TIBTimer.Timer 205
IBDatabase.TIBTransaction 206
IBDatabase.TIBTransaction.Active 215
IBDatabase.TIBTransaction.AddDatabase 209

IBDatabase.TIBTransaction.AutoStopAction 215
IBDatabase.TIBTransaction.Call 209
IBDatabase.TIBTransaction.CheckAutoStop 210
IBDatabase.TIBTransaction.CheckDatabasesInList 210
IBDatabase.TIBTransaction.CheckInTransaction 210
IBDatabase.TIBTransaction.CheckNotInTransaction 210
IBDatabase.TIBTransaction.Commit 211
IBDatabase.TIBTransaction.CommitRetaining 211
IBDatabase.TIBTransaction.Create 208
IBDatabase.TIBTransaction.DatabaseCount 216
IBDatabase.TIBTransaction.Databases 216
IBDatabase.TIBTransaction.DefaultAction 216
IBDatabase.TIBTransaction.DefaultDatabase 217
IBDatabase.TIBTransaction.Destroy 208
IBDatabase.TIBTransaction.FindDatabase 211
IBDatabase.TIBTransaction.FindDefaultDatabase 212
IBDatabase.TIBTransaction.Handle 217
IBDatabase.TIBTransaction.HandleIsShared 217
IBDatabase.TIBTransaction.IdleTimer 217
IBDatabase.TIBTransaction.InTransaction 218
IBDatabase.TIBTransaction.Loaded 212
IBDatabase.TIBTransaction.Notification 212
IBDatabase.TIBTransaction.OnIdleTimer 219
IBDatabase.TIBTransaction.Params 218
IBDatabase.TIBTransaction.RemoveDatabase 213
IBDatabase.TIBTransaction.RemoveDatabases 213
IBDatabase.TIBTransaction.Rollback 213
IBDatabase.TIBTransaction.RollbackRetaining 214
IBDatabase.TIBTransaction.SetHandle 214
IBDatabase.TIBTransaction.SQLObjectCount 218
IBDatabase.TIBTransaction.SQLObjects 218
IBDatabase.TIBTransaction.StartTransaction 214
IBDatabase.TIBTransaction.TPB 219
IBDatabase.TIBTransaction.TPBLength 219
IBDatabase.TPBConstantNames 230
IBDatabase.TPBConstantNames constant 230
IBDatabase.TPBPrefix 230
IBDatabase.TPBPrefix constant 230
IBDatabase.TTransactionAction 226
IBDatabase.TTransactionAction enumeration 226
IBDatabaseInfo 231
IBDatabaseInfo namespace 231
Classes 232
IBDatabaseInfo.TIBDatabaseInfo 232
IBDatabaseInfo.TIBDatabaseInfo.Allocation 247
IBDatabaseInfo.TIBDatabaseInfo.BackoutCount 247
IBDatabaseInfo.TIBDatabaseInfo.BaseLevel 248
IBDatabaseInfo.TIBDatabaseInfo.Call 238
IBDatabaseInfo.TIBDatabaseInfo.Create 234
IBDatabaseInfo.TIBDatabaseInfo.CurrentMemory 248
IBDatabaseInfo.TIBDatabaseInfo.Database 248
IBDatabaseInfo.TIBDatabaseInfo.DBFileName 248
IBDatabaseInfo.TIBDatabaseInfo.DBImplementationClass 249
IBDatabaseInfo.TIBDatabaseInfo.DBImplementationNo 249
IBDatabaseInfo.TIBDatabaseInfo.DBSiteName 249
IBDatabaseInfo.TIBDatabaseInfo.DBSQLDialect 249
IBDatabaseInfo.TIBDatabaseInfo.DeleteCount 250
IBDatabaseInfo.TIBDatabaseInfo.Destroy 234
IBDatabaseInfo.TIBDatabaseInfo.ExpungeCount 250
IBDatabaseInfo.TIBDatabaseInfo.FBackoutCount 235
IBDatabaseInfo.TIBDatabaseInfo.FDatabase 235
IBDatabaseInfo.TIBDatabaseInfo.FDeleteCount 236
IBDatabaseInfo.TIBDatabaseInfo.Fetches 250
IBDatabaseInfo.TIBDatabaseInfo.FExpungeCount 236
IBDatabaseInfo.TIBDatabaseInfo.FIBLoaded 236
IBDatabaseInfo.TIBDatabaseInfo.FInsertCount 236
IBDatabaseInfo.TIBDatabaseInfo.ForcedWrites 250
IBDatabaseInfo.TIBDatabaseInfo.FPurgeCount 236
IBDatabaseInfo.TIBDatabaseInfo.FReadIdxCount 237
IBDatabaseInfo.TIBDatabaseInfo.FReadSeqCount 237
IBDatabaseInfo.TIBDatabaseInfo.FUpdateCount 237
IBDatabaseInfo.TIBDatabaseInfo.FUserNames 237
IBDatabaseInfo.TIBDatabaseInfo.GetAllocation 239
IBDatabaseInfo.TIBDatabaseInfo.GetBackoutCount 239
IBDatabaseInfo.TIBDatabaseInfo.GetBaseLevel 239
IBDatabaseInfo.TIBDatabaseInfo.GetCurrentMemory 239
IBDatabaseInfo.TIBDatabaseInfo.GetDBFileName 239
IBDatabaseInfo.TIBDatabaseInfo.GetDBImplementationClass 240

IBDatabaseInfo.TIBDatabaseInfo.GetDBImplementationNo 240
IBDatabaseInfo.TIBDatabaseInfo.GetDBSiteName 240
IBDatabaseInfo.TIBDatabaseInfo.GetDBSQLDialect 240
IBDatabaseInfo.TIBDatabaseInfo.GetDeleteCount 241
IBDatabaseInfo.TIBDatabaseInfo.GetExpungeCount 241
IBDatabaseInfo.TIBDatabaseInfo.GetFetches 241
IBDatabaseInfo.TIBDatabaseInfo.GetForcedWrites 241
IBDatabaseInfo.TIBDatabaseInfo.GetInsertCount 241
IBDatabaseInfo.TIBDatabaseInfo.GetLongDatabaseInfo 242
IBDatabaseInfo.TIBDatabaseInfo.GetMarks 242
IBDatabaseInfo.TIBDatabaseInfo.GetMaxMemory 242
IBDatabaseInfo.TIBDatabaseInfo.GetNoReserve 242
IBDatabaseInfo.TIBDatabaseInfo.GetNumBuffers 243
IBDatabaseInfo.TIBDatabaseInfo.GetODSMajorVersion 243
IBDatabaseInfo.TIBDatabaseInfo.GetODSMinorVersion 243
IBDatabaseInfo.TIBDatabaseInfo.GetOperationCounts 243
IBDatabaseInfo.TIBDatabaseInfo.GetPageSize 244
IBDatabaseInfo.TIBDatabaseInfo.GetPurgeCount 244
IBDatabaseInfo.TIBDatabaseInfo.GetReadIdxCount 244
IBDatabaseInfo.TIBDatabaseInfo.GetReadOnly 244
IBDatabaseInfo.TIBDatabaseInfo.GetReads 244
IBDatabaseInfo.TIBDatabaseInfo.GetReadSeqCount 245
IBDatabaseInfo.TIBDatabaseInfo.GetStringDatabaseInfo 245
IBDatabaseInfo.TIBDatabaseInfo.GetSweepInterval 245
IBDatabaseInfo.TIBDatabaseInfo.GetUpdateCount 245
IBDatabaseInfo.TIBDatabaseInfo.GetUserNames 246
IBDatabaseInfo.TIBDatabaseInfo.GetVersion 246
IBDatabaseInfo.TIBDatabaseInfo.GetWrites 246
IBDatabaseInfo.TIBDatabaseInfo.InsertCount 251
IBDatabaseInfo.TIBDatabaseInfo.Marks 251
IBDatabaseInfo.TIBDatabaseInfo.MaxMemory 251
IBDatabaseInfo.TIBDatabaseInfo.NoReserve 251
IBDatabaseInfo.TIBDatabaseInfo.NumBuffers 252
IBDatabaseInfo.TIBDatabaseInfo.ODSMajorVersion 252
IBDatabaseInfo.TIBDatabaseInfo.ODSMinorVersion 252
IBDatabaseInfo.TIBDatabaseInfo.PageSize 252
IBDatabaseInfo.TIBDatabaseInfo.PurgeCount 253
IBDatabaseInfo.TIBDatabaseInfo.ReadIdxCount 253
IBDatabaseInfo.TIBDatabaseInfo.ReadOnly 253
IBDatabaseInfo.TIBDatabaseInfo.Reads 253
IBDatabaseInfo.TIBDatabaseInfo.ReadSeqCount 254
IBDatabaseInfo.TIBDatabaseInfo.SweepInterval 254
IBDatabaseInfo.TIBDatabaseInfo.UpdateCount 254
IBDatabaseInfo.TIBDatabaseInfo.UserNames 255
IBDatabaseInfo.TIBDatabaseInfo.Version 255
IBDatabaseInfo.TIBDatabaseInfo.Writes 255
IBErrorCode
    EIBError class 5
IBEvents 257
IBEvents namespace 257
    Classes 258
    Types 266
IBEvents.TErrorEvent 266
IBEvents.TErrorEvent type 266
IBEvents.TEventAlert 266
IBEvents.TEventAlert type 266
IBEvents.TIBEvents 258
IBEvents.TIBEvents.AutoRegister 263
IBEvents.TIBEvents.Create 259
IBEvents.TIBEvents.Database 263
IBEvents.TIBEvents.Destroy 260
IBEvents.TIBEvents.EventChange 260
IBEvents.TIBEvents.Events 264
IBEvents.TIBEvents.GetAutoRegister 260
IBEvents.TIBEvents.GetNativeHandle 261
IBEvents.TIBEvents.NativeHandle 264
IBEvents.TIBEvents.Notification 261
IBEvents.TIBEvents.OnError 265
IBEvents.TIBEvents.OnEventAlert 265
IBEvents.TIBEvents.Registered 264
IBEvents.TIBEvents.RegisterEvents 261
IBEvents.TIBEvents.SetAutoRegister 262
IBEvents.TIBEvents.ThreadEnded 262
IBEvents.TIBEvents.UnRegisterEvents 262
IBEvents.TIBEvents.ValidateDatabase 263
IBExtract 269
IBExtract namespace 269
    Classes 271

Constants 289
Structs, Records, Enums 283
Types 287
IBExtract.ColumnTypes 290
IBExtract.ColumnTypes constant 290
IBExtract.FILE_cache 290
IBExtract.FILE_cache constant 290
IBExtract.FILE_conditional 290
IBExtract.FILE_conditional constant 290
IBExtract.FILE_inactive 291
IBExtract.FILE_inactive constant 291
IBExtract.FILE_manual 291
IBExtract.FILE_manual constant 291
IBExtract.FILE_shadow 291
IBExtract.FILE_shadow constant 291
IBExtract.IntegralSubtypes 292
IBExtract.IntegralSubtypes constant 292
IBExtract.LOG_default 292
IBExtract.LOG_default constant 292
IBExtract.LOG_overflow 292
IBExtract.LOG_overflow constant 292
IBExtract.LOG_raw 293
IBExtract.LOG_raw constant 293
IBExtract.LOG_serial 293
IBExtract.LOG_serial constant 293
IBExtract.MAX_INTSUBTYPES 293
IBExtract.MAX_INTSUBTYPES constant 293
IBExtract.MAXSUBTYPES 294
IBExtract.MAXSUBTYPES constant 294
IBExtract.obj_computed 294
IBExtract.obj_computed constant 294
IBExtract.obj_count 294
IBExtract.obj_count constant 294
IBExtract.obj_exception 295
IBExtract.obj_exception constant 295
IBExtract.obj_expression_index 295
IBExtract.obj_expression_index constant 295
IBExtract.obj_field 295
IBExtract.obj_field constant 295
IBExtract.obj_index 296
IBExtract.obj_index constant 296
IBExtract.obj_procedure 296
IBExtract.obj_procedure constant 296
IBExtract.obj_relation 296
IBExtract.obj_relation constant 296
IBExtract.obj_sql_role 297
IBExtract.obj_sql_role constant 297
IBExtract.obj_trigger 297
IBExtract.obj_trigger constant 297
IBExtract.obj_user 297
IBExtract.obj_user constant 297
IBExtract.obj_user_group 298
IBExtract.obj_user_group constant 298
IBExtract.obj_validation 298
IBExtract.obj_validation constant 298
IBExtract.obj_view 298
IBExtract.obj_view constant 298
IBExtract.ODS_VERSION10 299
IBExtract.ODS_VERSION10 constant 299
IBExtract.ODS_VERSION6 299
IBExtract.ODS_VERSION6 constant 299
IBExtract.ODS_VERSION7 299
IBExtract.ODS_VERSION7 constant 299
IBExtract.ODS_VERSION8 300
IBExtract.ODS_VERSION8 constant 300
IBExtract.ODS_VERSION9 300
IBExtract.ODS_VERSION9 constant 300
IBExtract.priv_DELETE 300
IBExtract.priv_DELETE constant 300
IBExtract.priv_EXECUTE 301
IBExtract.priv_EXECUTE constant 301
IBExtract.priv_INSERT 301
IBExtract.priv_INSERT constant 301
IBExtract.priv_REFERENCES 301
IBExtract.priv_REFERENCES constant 301
IBExtract.priv_SELECT 302
IBExtract.priv_SELECT constant 302
IBExtract.priv_UNKNOWN 302

IBExtract.priv_UNKNOWN constant 302
IBExtract.priv_UPDATE 302
IBExtract.priv_UPDATE constant 302
IBExtract.PrivTypes 303
IBExtract.PrivTypes constant 303
IBExtract.SubTypes 303
IBExtract.SubTypes constant 303
IBExtract.TExtractObjectTypes 283
IBExtract.TExtractObjectTypes enumeration 283
IBExtract.TExtractType 284
IBExtract.TExtractType enumeration 284
IBExtract.TExtractTypes 287
IBExtract.TExtractTypes type 287
IBExtract.TIBExtract 271
IBExtract.TIBExtract.Create 272
IBExtract.TIBExtract.Database 281
IBExtract.TIBExtract.DatabaseInfo 281
IBExtract.TIBExtract.Destroy 273
IBExtract.TIBExtract.ExtractDDL 274
IBExtract.TIBExtract.ExtractListTable 274
IBExtract.TIBExtract.ExtractListView 274
IBExtract.TIBExtract.ExtractObject 274
IBExtract.TIBExtract.GetArrayField 276
IBExtract.TIBExtract.GetCharacterSets 276
IBExtract.TIBExtract.GetFieldType 276
IBExtract.TIBExtract.Items 281
IBExtract.TIBExtract.ListAllTables 276
IBExtract.TIBExtract.ListCheck 277
IBExtract.TIBExtract.ListCreateDb 277
IBExtract.TIBExtract.ListData 277
IBExtract.TIBExtract.ListDomains 277
IBExtract.TIBExtract.ListException 278
IBExtract.TIBExtract.ListFilters 278
IBExtract.TIBExtract.ListForeign 278
IBExtract.TIBExtract.ListFunctions 278
IBExtract.TIBExtract.ListGenerators 278
IBExtract.TIBExtract.ListGrants 279
IBExtract.TIBExtract.ListIndex 279
IBExtract.TIBExtract.ListProcs 279
IBExtract.TIBExtract.ListRoles 279
IBExtract.TIBExtract.ListTriggers 280
IBExtract.TIBExtract.ListViews 280
IBExtract.TIBExtract.Notification 280
IBExtract.TIBExtract.PrintSet 280
IBExtract.TIBExtract.ShowSystem 282
IBExtract.TIBExtract.Transaction 282
IBExtract.TPrivTypes 285
IBExtract.TPrivTypes record 285
IBExtract.TriggerTypes 303
IBExtract.TriggerTypes constant 303
IBExtract.TSQLType 286
IBExtract.TSQLType record 286
IBExtract.TSQLTypes 288
IBExtract.TSQLTypes type 288
IBInstall 305
IBInstall namespace 305
    Classes 307
    Structs, Records, Enums 328
    Types 333
IBInstall.EIBInstall 307
IBInstall.EIBInstall.Create 308
IBInstall.EIBInstall.InstallerError 308
IBInstall.EIBInstall.InstallError 309
IBInstall.EIBInstallerError 309
IBInstall.EIBInstallError 310
IBInstall.TCmdOption 328
IBInstall.TCmdOption enumeration 328
IBInstall.TCmdOptions 333
IBInstall.TCmdOptions type 333
IBInstall.TConnectivityOption 329
IBInstall.TConnectivityOption enumeration 329
IBInstall.TConnectivityOptions 333
IBInstall.TConnectivityOptions type 333
IBInstall.TErrorResult 329
IBInstall.TErrorResult enumeration 329
IBInstall.TExamplesOption 330
IBInstall.TExamplesOption enumeration 330
IBInstall.TExamplesOptions 334

IBInstall.TExamplesOptions type 334
IBInstall.TIBInstall 310
IBInstall.TIBInstall.Create 311
IBInstall.TIBInstall.DestinationDirectory 315
IBInstall.TIBInstall.Destroy 312
IBInstall.TIBInstall.GetOptionDescription 312
IBInstall.TIBInstall.GetOptionName 313
IBInstall.TIBInstall.GetOptionSpaceRequired 314
IBInstall.TIBInstall.InstallCheck 314
IBInstall.TIBInstall.InstallExecute 314
IBInstall.TIBInstall.InstallOptions 315
IBInstall.TIBInstall.SourceDirectory 316
IBInstall.TIBInstall.SuggestedDestination 316
IBInstall.TIBInstall.UnInstallFile 316
IBInstall.TIBInstallerError 330
IBInstall.TIBInstallerError enumeration 330
IBInstall.TIBSetup 317
IBInstall.TIBSetup.Call 318
IBInstall.TIBSetup.Create 318
IBInstall.TIBSetup.ErrorContext 319
IBInstall.TIBSetup.ErrorInternal 318
IBInstall.TIBSetup.GetInstallMessage 319
IBInstall.TIBSetup.IBInstallError 319
IBInstall.TIBSetup.MsgFilePath 320
IBInstall.TIBSetup.OnError 321
IBInstall.TIBSetup.OnStatusChange 322
IBInstall.TIBSetup.OnWarning 322
IBInstall.TIBSetup.Progress 320
IBInstall.TIBSetup.RebootToComplete 320
IBInstall.TIBSetup.StatusContext 321
IBInstall.TIBSetup.StatusInternal 319
IBInstall.TIBSetupOnError 334
IBInstall.TIBSetupOnError type 334
IBInstall.TIBSetupOnStatus 334
IBInstall.TIBSetupOnStatus type 334
IBInstall.TIBSetupOnWarning 335
IBInstall.TIBSetupOnWarning type 335
IBInstall.TIBUnInstall 322
IBInstall.TIBUnInstall.UnInstallCheck 324
IBInstall.TIBUnInstall.UnInstallExecute 324
IBInstall.TIBUnInstall.UnInstallFile 324
IBInstall.TInstallOptions 325
IBInstall.TInstallOptions.CmdLineTools 325
IBInstall.TInstallOptions.ConnectivityClients 326
IBInstall.TInstallOptions.Examples 326
IBInstall.TInstallOptions.MainComponents 326
IBInstall.TIscError 335
IBInstall.TIscError type 335
IBInstall.TMainOption 331
IBInstall.TMainOption enumeration 331
IBInstall.TMainOptions 336
IBInstall.TMainOptions type 336
IBInstall.TStatusResult 332
IBInstall.TStatusResult enumeration 332
IBInstall.TWarningResult 332
IBInstall.TWarningResult enumeration 332
IBInstallError
    TIBSetup class 319
IBIntf 337
IBIntf namespace 337
    Functions 340
    Variables 351
IBIntf.BLOB_get 352
IBIntf.BLOB_get variable 352
IBIntf.BLOB_put 352
IBIntf.BLOB_put variable 352
IBIntf.CheckIBInstallLoaded 340
IBIntf.CheckIBInstallLoaded function 340
IBIntf.CheckIBLoaded 341
IBIntf.CheckIBLoaded function 341
IBIntf.FreeIBInstallLibrary 341
IBIntf.FreeIBInstallLibrary function 341
IBIntf.FreeIBLibrary 341
IBIntf.FreeIBLibrary function 341
IBIntf.GetIBClientVersion 342
IBIntf.GetIBClientVersion function 342
IBIntf.isc_add_user 353
IBIntf.isc_add_user variable 353

IBIntf.isc_attach_database 353
IBIntf.isc_attach_database variable 353
IBIntf.isc_blob_info 353
IBIntf.isc_blob_info variable 353
IBIntf.isc_cancel_events 354
IBIntf.isc_cancel_events variable 354
IBIntf.isc_close_blob 354
IBIntf.isc_close_blob variable 354
IBIntf.isc_commit_retaining 354
IBIntf.isc_commit_retaining variable 354
IBIntf.isc_commit_transaction 355
IBIntf.isc_commit_transaction variable 355
IBIntf.isc_create_blob2 355
IBIntf.isc_create_blob2 variable 355
IBIntf.isc_database_info 355
IBIntf.isc_database_info variable 355
IBIntf.isc_decode_date 356
IBIntf.isc_decode_date variable 356
IBIntf.isc_decode_sql_date 356
IBIntf.isc_decode_sql_date variable 356
IBIntf.isc_decode_sql_date_stub 342
IBIntf.isc_decode_sql_date_stub function 342
IBIntf.isc_decode_sql_time 356
IBIntf.isc_decode_sql_time variable 356
IBIntf.isc_decode_sql_time_stub 342
IBIntf.isc_decode_sql_time_stub function 342
IBIntf.isc_decode_timestamp 357
IBIntf.isc_decode_timestamp variable 357
IBIntf.isc_decode_timestamp_stub 343
IBIntf.isc_decode_timestamp_stub function 343
IBIntf.isc_delete_user 357
IBIntf.isc_delete_user variable 357
IBIntf.isc_detach_database 357
IBIntf.isc_detach_database variable 357
IBIntf.isc_drop_database 358
IBIntf.isc_drop_database variable 358
IBIntf.isc_dsql_alloc_statement2 358
IBIntf.isc_dsql_alloc_statement2 variable 358
IBIntf.isc_dsql_describe 358
IBIntf.isc_dsql_describe variable 358
IBIntf.isc_dsql_describe_bind 359
IBIntf.isc_dsql_describe_bind variable 359
IBIntf.isc_dsql_execute 359
IBIntf.isc_dsql_execute variable 359
IBIntf.isc_dsql_execute_immediate 359
IBIntf.isc_dsql_execute_immediate variable 359
IBIntf.isc_dsql_execute2 360
IBIntf.isc_dsql_execute2 variable 360
IBIntf.isc_dsql_fetch 360
IBIntf.isc_dsql_fetch variable 360
IBIntf.isc_dsql_free_statement 360
IBIntf.isc_dsql_free_statement variable 360
IBIntf.isc_dsql_prepare 361
IBIntf.isc_dsql_prepare variable 361
IBIntf.isc_dsql_set_cursor_name 361
IBIntf.isc_dsql_set_cursor_name variable 361
IBIntf.isc_dsql_sql_info 361
IBIntf.isc_dsql_sql_info variable 361
IBIntf.isc_encode_date 362
IBIntf.isc_encode_date variable 362
IBIntf.isc_encode_sql_date 362
IBIntf.isc_encode_sql_date variable 362
IBIntf.isc_encode_sql_date_stub 343
IBIntf.isc_encode_sql_date_stub function 343
IBIntf.isc_encode_sql_time 362
IBIntf.isc_encode_sql_time variable 362
IBIntf.isc_encode_sql_time_stub 343
IBIntf.isc_encode_sql_time_stub function 343
IBIntf.isc_encode_timestamp 363
IBIntf.isc_encode_timestamp variable 363
IBIntf.isc_encode_timestamp_stub 344
IBIntf.isc_encode_timestamp_stub function 344
IBIntf.isc_event_block 363
IBIntf.isc_event_block variable 363
IBIntf.isc_event_counts 363
IBIntf.isc_event_counts variable 363
IBIntf.isc_free 364
IBIntf.isc_free variable 364

IBIntf.isc_get_segment 364
IBIntf.isc_get_segment variable 364
IBIntf.isc_install_clear_options 364
IBIntf.isc_install_clear_options variable 364
IBIntf.isc_install_clear_options_stub 344
IBIntf.isc_install_clear_options_stub function 344
IBIntf.isc_install_execute 365
IBIntf.isc_install_execute variable 365
IBIntf.isc_install_execute_stub 344
IBIntf.isc_install_execute_stub function 344
IBIntf.isc_install_get_info 365
IBIntf.isc_install_get_info variable 365
IBIntf.isc_install_get_info_stub 345
IBIntf.isc_install_get_info_stub function 345
IBIntf.isc_install_get_message 365
IBIntf.isc_install_get_message variable 365
IBIntf.isc_install_get_message_stub 345
IBIntf.isc_install_get_message_stub function 345
IBIntf.isc_install_load_external_text 366
IBIntf.isc_install_load_external_text variable 366
IBIntf.isc_install_load_external_text_stub 345
IBIntf.isc_install_load_external_text_stub function 345
IBIntf.isc_install_precheck 366
IBIntf.isc_install_precheck variable 366
IBIntf.isc_install_precheck_stub 346
IBIntf.isc_install_precheck_stub function 346
IBIntf.isc_install_set_option 366
IBIntf.isc_install_set_option variable 366
IBIntf.isc_install_set_option_stub 346
IBIntf.isc_install_set_option_stub function 346
IBIntf.isc_install_unset_option 367
IBIntf.isc_install_unset_option variable 367
IBIntf.isc_install_unset_option_stub 346
IBIntf.isc_install_unset_option_stub function 346
IBIntf.isc_interprete 367
IBIntf.isc_interprete variable 367
IBIntf.isc_modify_user 367
IBIntf.isc_modify_user variable 367
IBIntf.isc_open_blob2 368
IBIntf.isc_open_blob2 variable 368
IBIntf.isc_prepare_transaction 368
IBIntf.isc_prepare_transaction variable 368
IBIntf.isc_prepare_transaction2 368
IBIntf.isc_prepare_transaction2 variable 368
IBIntf.isc_put_segment 369
IBIntf.isc_put_segment variable 369
IBIntf.isc_que_events 369
IBIntf.isc_que_events variable 369
IBIntf.isc_rollback_retaining 369
IBIntf.isc_rollback_retaining variable 369
IBIntf.isc_rollback_retaining_stub 347
IBIntf.isc_rollback_retaining_stub function 347
IBIntf.isc_rollback_transaction 370
IBIntf.isc_rollback_transaction variable 370
IBIntf.isc_service_attach 370
IBIntf.isc_service_attach variable 370
IBIntf.isc_service_attach_stub 347
IBIntf.isc_service_attach_stub function 347
IBIntf.isc_service_detach 370
IBIntf.isc_service_detach variable 370
IBIntf.isc_service_detach_stub 347
IBIntf.isc_service_detach_stub function 347
IBIntf.isc_service_query 371
IBIntf.isc_service_query variable 371
IBIntf.isc_service_query_stub 348
IBIntf.isc_service_query_stub function 348
IBIntf.isc_service_start 371
IBIntf.isc_service_start variable 371
IBIntf.isc_service_start_stub 348
IBIntf.isc_service_start_stub function 348
IBIntf.isc_sql_interprete 371
IBIntf.isc_sql_interprete variable 371
IBIntf.isc_sqlcode 372
IBIntf.isc_sqlcode variable 372
IBIntf.isc_start_multiple 372
IBIntf.isc_start_multiple variable 372
IBIntf.isc_uninstall_execute 372
IBIntf.isc_uninstall_execute variable 372

IBIntf.isc_uninstall_execute_stub 348
IBIntf.isc_uninstall_execute_stub function 348
IBIntf.isc_uninstall_precheck 373
IBIntf.isc_uninstall_precheck variable 373
IBIntf.isc_uninstall_precheck_stub 349
IBIntf.isc_uninstall_precheck_stub function 349
IBIntf.isc_vax_integer 373
IBIntf.isc_vax_integer variable 373
IBIntf.LoadIBInstallLibrary 349
IBIntf.LoadIBInstallLibrary function 349
IBIntf.LoadIBLibrary 349
IBIntf.LoadIBLibrary function 349
IBIntf.TryIBLoad 350
IBIntf.TryIBLoad function 350
IBOA interface 2793
about IBOA interface 2793
Deactivate 2794
GetPrincipal 2794
GetPrincipalLength 2794
IBOA Members 2793
IBOA methods 2794
ImplIsReady 2794
ObjIsReady 2795
IBQuery 375
IBQuery namespace 375
Classes 376
IBQuery.TIBQuery 376
IBQuery.TIBQuery.Active 387
IBQuery.TIBQuery.AfterDatabaseDisconnect 394
IBQuery.TIBQuery.AfterTransactionEnd 394
IBQuery.TIBQuery.BatchInput 382
IBQuery.TIBQuery.BatchOutput 382
IBQuery.TIBQuery.BeforeDatabaseDisconnect 394
IBQuery.TIBQuery.BeforeTransactionEnd 395
IBQuery.TIBQuery.BufferChunks 387
IBQuery.TIBQuery.CachedUpdates 388
IBQuery.TIBQuery.Constraints 388
IBQuery.TIBQuery.Create 381
IBQuery.TIBQuery.DatabaseFree 395
IBQuery.TIBQuery.DataSource 388
IBQuery.TIBQuery.DefineProperties 382
IBQuery.TIBQuery.Destroy 381
IBQuery.TIBQuery.Disconnect 383
IBQuery.TIBQuery.ExecSQL 383
IBQuery.TIBQuery.Filtered 389
IBQuery.TIBQuery.GenerateParamNames 389
IBQuery.TIBQuery.GenerateQueryForLiveUpdate 383
IBQuery.TIBQuery.GeneratorField 389
IBQuery.TIBQuery.GetDetailLinkFields 383
IBQuery.TIBQuery.GetParamsCount 384
IBQuery.TIBQuery.InitFieldDefs 384
IBQuery.TIBQuery.InternalOpen 384
IBQuery.TIBQuery.LiveMode 389
IBQuery.TIBQuery.OnFilterRecord 390
IBQuery.TIBQuery.ParamByName 384
IBQuery.TIBQuery.ParamCheck 390
IBQuery.TIBQuery.ParamCount 390
IBQuery.TIBQuery.Params 391
IBQuery.TIBQuery.Prepare 385
IBQuery.TIBQuery.Prepared 391
IBQuery.TIBQuery.PSExecute 385
IBQuery.TIBQuery.PSGetParams 385
IBQuery.TIBQuery.PSGetTableName 385
IBQuery.TIBQuery.PSSetCommandText 386
IBQuery.TIBQuery.PSSetParams 386
IBQuery.TIBQuery.RowsAffected 391
IBQuery.TIBQuery.SetFiltered 386
IBQuery.TIBQuery.SQL 392
IBQuery.TIBQuery.StatementType 392
IBQuery.TIBQuery.StmtHandle 392
IBQuery.TIBQuery.Text 392
IBQuery.TIBQuery.TransactionFree 395
IBQuery.TIBQuery.UniDirectional 393
IBQuery.TIBQuery.UnPrepare 386
IBQuery.TIBQuery.UpdateObject 393
IBServices 397
IBServices namespace 397
Classes 399

Constants 491
Structs, Records, Enums 477
Types 486
IBServices.DefaultBufferSize 491
IBServices.DefaultBufferSize constant 491
IBServices.SPBConstantNames 491
IBServices.SPBConstantNames constant 491
IBServices.SPBConstantValues 492
IBServices.SPBConstantValues constant 492
IBServices.SPBPrefix 492
IBServices.SPBPrefix constant 492
IBServices.TBackupOption 477
IBServices.TBackupOption enumeration 477
IBServices.TBackupOptions 486
IBServices.TBackupOptions type 486
IBServices.TConfigFileData 399
IBServices.TConfigFileData.ConfigFileKey 401
IBServices.TConfigFileData.ConfigFileValue 401
IBServices.TConfigFileData.Create 400
IBServices.TConfigFileData.Destroy 400
IBServices.TConfigParams 401
IBServices.TConfigParams.BaseLocation 402
IBServices.TConfigParams.ConfigFileData 403
IBServices.TConfigParams.ConfigFileParams 403
IBServices.TConfigParams.Create 402
IBServices.TConfigParams.Destroy 402
IBServices.TConfigParams.LockFileLocation 403
IBServices.TConfigParams.MessageFileLocation 403
IBServices.TConfigParams.SecurityDatabaseLocation 404
IBServices.TDatabaseInfo 404
IBServices.TDatabaseInfo.Create 405
IBServices.TDatabaseInfo.DbName 405
IBServices.TDatabaseInfo.Destroy 405
IBServices.TDatabaseInfo.NoOfAttachments 405
IBServices.TDatabaseInfo.NoOfDatabases 406
IBServices.TIBBackupRestoreService 406
IBServices.TIBBackupRestoreService.Verbose 408
IBServices.TIBBackupService 408
IBServices.TIBBackupService.BackupFile 411
IBServices.TIBBackupService.BlockingFactor 411
IBServices.TIBBackupService.Create 410
IBServices.TIBBackupService.DatabaseName 411
IBServices.TIBBackupService.Destroy 410
IBServices.TIBBackupService.Options 411
IBServices.TIBBackupService.SetServiceStartOptions 410
IBServices.TIBConfigService 412
IBServices.TIBConfigService.ActivateShadow 414
IBServices.TIBConfigService.BringDatabaseOnline 414
IBServices.TIBConfigService.DatabaseName 417
IBServices.TIBConfigService.ServiceStart 414
IBServices.TIBConfigService.SetAsyncMode 415
IBServices.TIBConfigService.SetDBSqlDialect 415
IBServices.TIBConfigService.SetPageBuffers 415
IBServices.TIBConfigService.SetReadOnly 416
IBServices.TIBConfigService.SetReserveSpace 416
IBServices.TIBConfigService.SetSweepInterval 416
IBServices.TIBConfigService.ShutdownDatabase 417
IBServices.TIBControlAndQueryService 417
IBServices.TIBControlAndQueryService.Action 420
IBServices.TIBControlAndQueryService.BufferSize 420
IBServices.TIBControlAndQueryService.Create 419
IBServices.TIBControlAndQueryService.Eof 421
IBServices.TIBControlAndQueryService.GetNextChunk 419
IBServices.TIBControlAndQueryService.GetNextLine 420
IBServices.TIBControlService 421
IBServices.TIBControlService.Create 422
IBServices.TIBControlService.InternalServiceStart 423
IBServices.TIBControlService.IsServiceRunning 424
IBServices.TIBControlService.ServiceStart 423
IBServices.TIBControlService.ServiceStartAddParam 424
IBServices.TIBControlService.ServiceStartParams 425
IBServices.TIBControlService.SetServiceStartOptions 424
IBServices.TIBCustomService 425
IBServices.TIBCustomService.Active 430
IBServices.TIBCustomService.Attach 427
IBServices.TIBCustomService.BufferSize 430
IBServices.TIBCustomService.CheckActive 427
IBServices.TIBCustomService.CheckInactive 428

IBServices.TIBCustomService.Create 426
IBServices.TIBCustomService.Destroy 426
IBServices.TIBCustomService.Detach 428
IBServices.TIBCustomService.Handle 430
IBServices.TIBCustomService.InternalServiceQuery 428
IBServices.TIBCustomService.Loaded 428
IBServices.TIBCustomService.Login 429
IBServices.TIBCustomService.LoginPrompt 430
IBServices.TIBCustomService.OnAttach 433
IBServices.TIBCustomService.OnLogin 433
IBServices.TIBCustomService.OutputBuffer 431
IBServices.TIBCustomService.OutputBufferOption 431
IBServices.TIBCustomService.Params 431
IBServices.TIBCustomService.Protocol 431
IBServices.TIBCustomService.ServerName 431
IBServices.TIBCustomService.ServiceParamBySPB 432
IBServices.TIBCustomService.ServiceQueryParams 432
IBServices.TIBCustomService.TraceFlags 432
IBServices.TIBLicensingService 433
IBServices.TIBLicensingService.Action 436
IBServices.TIBLicensingService.AddLicense 435
IBServices.TIBLicensingService.ID 436
IBServices.TIBLicensingService.Key 437
IBServices.TIBLicensingService.RemoveLicense 435
IBServices.TIBLicensingService.SetServiceStartOptions 436
IBServices.TIBLogService 437
IBServices.TIBLogService.SetServiceStartOptions 439
IBServices.TIBRestoreService 439
IBServices.TIBRestoreService.BackupFile 442
IBServices.TIBRestoreService.Create 440
IBServices.TIBRestoreService.DatabaseName 442
IBServices.TIBRestoreService.Destroy 441
IBServices.TIBRestoreService.Options 442
IBServices.TIBRestoreService.PageBuffers 442
IBServices.TIBRestoreService.PageSize 443
IBServices.TIBRestoreService.SetServiceStartOptions 441
IBServices.TIBSecurityService 443
IBServices.TIBSecurityService.AddUser 446
IBServices.TIBSecurityService.Create 445
IBServices.TIBSecurityService.DeleteUser 446
IBServices.TIBSecurityService.Destroy 445
IBServices.TIBSecurityService.DisplayUser 446
IBServices.TIBSecurityService.DisplayUsers 447
IBServices.TIBSecurityService.FirstName 449
IBServices.TIBSecurityService.GroupID 449
IBServices.TIBSecurityService.LastName 449
IBServices.TIBSecurityService.Loaded 447
IBServices.TIBSecurityService.MiddleName 449
IBServices.TIBSecurityService.ModifyUser 448
IBServices.TIBSecurityService.Password 450
IBServices.TIBSecurityService.SecurityAction 450
IBServices.TIBSecurityService.SetServiceStartOptions 448
IBServices.TIBSecurityService.SQlRole 450
IBServices.TIBSecurityService.UserID 450
IBServices.TIBSecurityService.UserInfo 451
IBServices.TIBSecurityService.UserInfoCount 451
IBServices.TIBSecurityService.UserName 451
IBServices.TIBServerProperties 452
IBServices.TIBServerProperties.ConfigParams 456
IBServices.TIBServerProperties.Create 453
IBServices.TIBServerProperties.DatabaseInfo 456
IBServices.TIBServerProperties.Destroy 453
IBServices.TIBServerProperties.Fetch 454
IBServices.TIBServerProperties.FetchConfigParams 454
IBServices.TIBServerProperties.FetchDatabaseInfo 455
IBServices.TIBServerProperties.FetchLicenseInfo 455
IBServices.TIBServerProperties.FetchLicenseMaskInfo 455
IBServices.TIBServerProperties.FetchVersionInfo 455
IBServices.TIBServerProperties.LicenseInfo 457
IBServices.TIBServerProperties.LicenseMaskInfo 457
IBServices.TIBServerProperties.Options 457
IBServices.TIBServerProperties.VersionInfo 457
IBServices.TIBStatisticalService 458
IBServices.TIBStatisticalService.DatabaseName 460
IBServices.TIBStatisticalService.Options 460
IBServices.TIBStatisticalService.SetServiceStartOptions 460
IBServices.TIBValidationService 461
IBServices.TIBValidationService.Create 462

IBServices.TIBValidationService.DatabaseName 464
IBServices.TIBValidationService.Destroy 463
IBServices.TIBValidationService.FetchLimboTransactionInfo 463
IBServices.TIBValidationService.FixLimboTransactionErrors 463
IBServices.TIBValidationService.GlobalAction 465
IBServices.TIBValidationService.LimboTransactionInfo 465
IBServices.TIBValidationService.LimboTransactionInfoCount 465
IBServices.TIBValidationService.Options 465
IBServices.TIBValidationService.SetServiceStartOptions 464
IBServices.TLicenseInfo 466
IBServices.TLicenseInfo.Create 467
IBServices.TLicenseInfo.Desc 467
IBServices.TLicenseInfo.Destroy 467
IBServices.TLicenseInfo.Id 467
IBServices.TLicenseInfo.Key 468
IBServices.TLicenseInfo.LicensedUsers 468
IBServices.TLicenseMaskInfo 468
IBServices.TLicenseMaskInfo.CapabilityMask 469
IBServices.TLicenseMaskInfo.LicenseMask 469
IBServices.TLicensingAction 478
IBServices.TLicensingAction enumeration 478
IBServices.TLimboTransactionInfo 469
IBServices.TLimboTransactionInfo.Action 471
IBServices.TLimboTransactionInfo.Advise 471
IBServices.TLimboTransactionInfo.HostSite 471
IBServices.TLimboTransactionInfo.ID 471
IBServices.TLimboTransactionInfo.MultiDatabase 472
IBServices.TLimboTransactionInfo.RemoteDatabasePath 472
IBServices.TLimboTransactionInfo.RemoteSite 472
IBServices.TLimboTransactionInfo.State 472
IBServices.TLoginEvent 487
IBServices.TLoginEvent type 487
IBServices.TOutputBufferOption 478
IBServices.TOutputBufferOption enumeration 478
IBServices.TPropertyOption 479
IBServices.TPropertyOption enumeration 479
IBServices.TPropertyOptions 487
IBServices.TPropertyOptions type 487
IBServices.TProtocol 479
IBServices.TProtocol enumeration 479
IBServices.TRestoreOption 480
IBServices.TRestoreOption enumeration 480
IBServices.TRestoreOptions 488
IBServices.TRestoreOptions type 488
IBServices.TSecurityAction 480
IBServices.TSecurityAction enumeration 480
IBServices.TSecurityModifyParam 481
IBServices.TSecurityModifyParam enumeration 481
IBServices.TSecurityModifyParams 489
IBServices.TSecurityModifyParams type 489
IBServices.TShutdownMode 481
IBServices.TShutdownMode enumeration 481
IBServices.TStatOption 482
IBServices.TStatOption enumeration 482
IBServices.TStatOptions 489
IBServices.TStatOptions type 489
IBServices.TTransactionAction 482
IBServices.TTransactionAction enumeration 482
IBServices.TTransactionAdvise 483
IBServices.TTransactionAdvise enumeration 483
IBServices.TTransactionGlobalAction 483
IBServices.TTransactionGlobalAction enumeration 483
IBServices.TTransactionState 484
IBServices.TTransactionState enumeration 484
IBServices.TUserInfo 472
IBServices.TUserInfo.FirstName 474
IBServices.TUserInfo.GroupID 474
IBServices.TUserInfo.LastName 474
IBServices.TUserInfo.MiddleName 474
IBServices.TUserInfo.UserID 474
IBServices.TUserInfo.UserName 475
IBServices.TValidateOption 485
IBServices.TValidateOption enumeration 485
IBServices.TValidateOptions 490
IBServices.TValidateOptions type 490
IBServices.TVersionInfo 475
IBServices.TVersionInfo.ServerImplementation 476

IBServices.TVersionInfo.ServerVersion 476
IBServices.TVersionInfo.ServiceVersion 476
IBSQL 493
IBSQL namespace 493
Classes 494
Structs, Records, Enums 556
Types 558
IBSQL.TIBBatch 494
IBSQL.TIBBatch.Columns 496
IBSQL.TIBBatch.FColumns 495
IBSQL.TIBBatch.FFilename 495
IBSQL.TIBBatch.Filename 497
IBSQL.TIBBatch.FParams 496
IBSQL.TIBBatch.Params 497
IBSQL.TIBBatch.ReadyFile 496
IBSQL.TIBBatchInput 497
IBSQL.TIBBatchInput.ReadParameters 498
IBSQL.TIBBatchOutput 498
IBSQL.TIBBatchOutput.WriteColumns 499
IBSQL.TIBInputDelimitedFile 500
IBSQL.TIBInputDelimitedFile.ColDelimiter 504
IBSQL.TIBInputDelimitedFile.Destroy 501
IBSQL.TIBInputDelimitedFile.FColDelimiter 501
IBSQL.TIBInputDelimitedFile.FEOF 501
IBSQL.TIBInputDelimitedFile.FFile 502
IBSQL.TIBInputDelimitedFile.FLookAhead 502
IBSQL.TIBInputDelimitedFile.FReadBlanksAsNull 502
IBSQL.TIBInputDelimitedFile.FRowDelimiter 502
IBSQL.TIBInputDelimitedFile.FSkipTitles 503
IBSQL.TIBInputDelimitedFile.GetColumn 503
IBSQL.TIBInputDelimitedFile.ReadBlanksAsNull 504
IBSQL.TIBInputDelimitedFile.ReadParameters 503
IBSQL.TIBInputDelimitedFile.ReadyFile 503
IBSQL.TIBInputDelimitedFile.RowDelimiter 504
IBSQL.TIBInputDelimitedFile.SkipTitles 505
IBSQL.TIBInputRawFile 505
IBSQL.TIBInputRawFile.Destroy 506
IBSQL.TIBInputRawFile.FFile 506
IBSQL.TIBInputRawFile.ReadParameters 507
IBSQL.TIBInputRawFile.ReadyFile 507
IBSQL.TIBOutputDelimitedFile 507
IBSQL.TIBOutputDelimitedFile.ColDelimiter 510
IBSQL.TIBOutputDelimitedFile.Destroy 508
IBSQL.TIBOutputDelimitedFile.FColDelimiter 509
IBSQL.TIBOutputDelimitedFile.FFile 509
IBSQL.TIBOutputDelimitedFile.FOutputTitles 509
IBSQL.TIBOutputDelimitedFile.FRowDelimiter 509
IBSQL.TIBOutputDelimitedFile.OutputTitles 511
IBSQL.TIBOutputDelimitedFile.ReadyFile 510
IBSQL.TIBOutputDelimitedFile.RowDelimiter 511
IBSQL.TIBOutputDelimitedFile.WriteColumns 510
IBSQL.TIBOutputRawFile 511
IBSQL.TIBOutputRawFile.Destroy 512
IBSQL.TIBOutputRawFile.FFile 512
IBSQL.TIBOutputRawFile.ReadyFile 513
IBSQL.TIBOutputRawFile.WriteColumns 513
IBSQL.TIBSQL 513
IBSQL.TIBSQL.BatchInput 521
IBSQL.TIBSQL.BatchOutput 522
IBSQL.TIBSQL.BeforeTransactionEnd 522
IBSQL.TIBSQL.Bof 530
IBSQL.TIBSQL.Call 522
IBSQL.TIBSQL.CheckClosed 522
IBSQL.TIBSQL.CheckOpen 523
IBSQL.TIBSQL.CheckValidStatement 523
IBSQL.TIBSQL.Close 523
IBSQL.TIBSQL.Create 516
IBSQL.TIBSQL.Current 523
IBSQL.TIBSQL.Database 530
IBSQL.TIBSQL.DBHandle 530
IBSQL.TIBSQL.Destroy 516
IBSQL.TIBSQL.DoBeforeDatabaseDisconnect 524
IBSQL.TIBSQL.Eof 530
IBSQL.TIBSQL.ExecQuery 524
IBSQL.TIBSQL.FBase 517
IBSQL.TIBSQL.FBOF 517
IBSQL.TIBSQL.FCursor 517
IBSQL.TIBSQL.FEOF 517

IBSQL.TIBSQL.FGenerateParamNames 518
IBSQL.TIBSQL.FGoToFirstRecordOnExecute 518
IBSQL.TIBSQL.FHandle 518
IBSQL.TIBSQL.FieldByName 524
IBSQL.TIBSQL.FieldIndex 531
IBSQL.TIBSQL.Fields 531
IBSQL.TIBSQL.FOnSQLChanging 518
IBSQL.TIBSQL.FOpen 519
IBSQL.TIBSQL.FParamCheck 519
IBSQL.TIBSQL.FPrepared 519
IBSQL.TIBSQL.FProcessedSQL 519
IBSQL.TIBSQL.FRecordCount 519
IBSQL.TIBSQL.FreeHandle 524
IBSQL.TIBSQL.FSQL 520
IBSQL.TIBSQL.FSQLParams 520
IBSQL.TIBSQL.FSQLRecord 520
IBSQL.TIBSQL.FSQLType 520
IBSQL.TIBSQL.GenerateParamNames 531
IBSQL.TIBSQL.GetDatabase 525
IBSQL.TIBSQL.GetDBHandle 525
IBSQL.TIBSQL.GetEOF 525
IBSQL.TIBSQL.GetFieldIndex 525
IBSQL.TIBSQL.GetFields 525
IBSQL.TIBSQL.GetPlan 526
IBSQL.TIBSQL.GetRecordCount 526
IBSQL.TIBSQL.GetRowsAffected 526
IBSQL.TIBSQL.GetSQLParams 526
IBSQL.TIBSQL.GetTransaction 527
IBSQL.TIBSQL.GetTRHandle 527
IBSQL.TIBSQL.GetUniqueRelationName 527
IBSQL.TIBSQL.GoToFirstRecordOnExecute 532
IBSQL.TIBSQL.Handle 532
IBSQL.TIBSQL.Next 527
IBSQL.TIBSQL.OnSQLChanging 535
IBSQL.TIBSQL.Open 532
IBSQL.TIBSQL.ParamByName 527
IBSQL.TIBSQL.ParamCheck 532
IBSQL.TIBSQL.Params 533
IBSQL.TIBSQL.Plan 533
IBSQL.TIBSQL.Prepare 528
IBSQL.TIBSQL.Prepared 533
IBSQL.TIBSQL.PreprocessSQL 528
IBSQL.TIBSQL.RecordCount 533
IBSQL.TIBSQL.RowsAffected 534
IBSQL.TIBSQL.SetDatabase 528
IBSQL.TIBSQL.SetSQL 528
IBSQL.TIBSQL.SetTransaction 529
IBSQL.TIBSQL.SQL 534
IBSQL.TIBSQL.SQLChanging 529
IBSQL.TIBSQL.SQLType 534
IBSQL.TIBSQL.Transaction 535
IBSQL.TIBSQL.TRHandle 535
IBSQL.TIBSQL.UniqueRelationName 535
IBSQLTypes 556
IBSQLTypes enumeration 556
IBSQL.TIBXSQLDA 536
IBSQL.TIBXSQLDA.AddName 540
IBSQL.TIBXSQLDA.AsXSQLDA 542
IBSQL.TIBXSQLDA.ByName 540
IBSQL.TIBXSQLDA.Count 543
IBSQL.TIBXSQLDA.Create 537
IBSQL.TIBXSQLDA.Destroy 537
IBSQL.TIBXSQLDA.FCount 538
IBSQL.TIBXSQLDA.FNames 538
IBSQL.TIBXSQLDA.FSize 538
IBSQL.TIBXSQLDA.FSQL 539
IBSQL.TIBXSQLDA.FUniqueRelationName 539
IBSQL.TIBXSQLDA.FXSQLDA 539
IBSQL.TIBXSQLDA.FXSQLVARs 539
IBSQL.TIBXSQLDA.GetModified 540
IBSQL.TIBXSQLDA.GetNames 541
IBSQL.TIBXSQLDA.GetRecordSize 541
IBSQL.TIBXSQLDA.GetXSQLDA 541
IBSQL.TIBXSQLDA.GetXSQLVAR 541
IBSQL.TIBXSQLDA.GetXSQLVARByName 541
IBSQL.TIBXSQLDA.Initialize 542
IBSQL.TIBXSQLDA.Modified 543
IBSQL.TIBXSQLDA.Names 543

IBSQL.TIBXSQLDA.RecordSize 543
IBSQL.TIBXSQLDA.SetCount 542
IBSQL.TIBXSQLDA.UniqueRelationName 544
IBSQL.TIBXSQLDA.Vars 544
IBSQL.TIBXSQLVAR 544
IBSQL.TIBXSQLVAR.AsCurrency 548
IBSQL.TIBXSQLVAR.AsDate 548
IBSQL.TIBXSQLVAR.AsDateTime 549
IBSQL.TIBXSQLVAR.AsDouble 549
IBSQL.TIBXSQLVAR.AsFloat 549
IBSQL.TIBXSQLVAR.AsInt64 550
IBSQL.TIBXSQLVAR.AsInteger 550
IBSQL.TIBXSQLVAR.AsLong 550
IBSQL.TIBXSQLVAR.AsPointer 550
IBSQL.TIBXSQLVAR.AsQuad 551
IBSQL.TIBXSQLVAR.AsShort 551
IBSQL.TIBXSQLVAR.Assign 546
IBSQL.TIBXSQLVAR.AsString 551
IBSQL.TIBXSQLVAR.AsTime 551
IBSQL.TIBXSQLVAR.AsTrimString 552
IBSQL.TIBXSQLVAR.AsVariant 552
IBSQL.TIBXSQLVAR.AsXSQLVAR 552
IBSQL.TIBXSQLVAR.Clear 546
IBSQL.TIBXSQLVAR.Create 545
IBSQL.TIBXSQLVAR.Data 553
IBSQL.TIBXSQLVAR.Index 553
IBSQL.TIBXSQLVAR.IsNull 553
IBSQL.TIBXSQLVAR.IsNullable 553
IBSQL.TIBXSQLVAR.LoadFromFile 547
IBSQL.TIBXSQLVAR.LoadFromStream 547
IBSQL.TIBXSQLVAR.Modified 554
IBSQL.TIBXSQLVAR.Name 554
IBSQL.TIBXSQLVAR.SaveToFile 547
IBSQL.TIBXSQLVAR.SaveToStream 547
IBSQL.TIBXSQLVAR.Size 554
IBSQL.TIBXSQLVAR.SQLType 554
IBSQL.TIBXSQLVAR.Value 555
IBSQL.TIBXSQLVARArray 558
IBSQL.TIBXSQLVARArray type 558
IBSQLMonitor 559
IBSQLMonitor namespace 559
Classes 561
Constants 578
Functions 575
Interfaces 567
Types 577
IBSQLMonitor.DisableMonitoring 575
IBSQLMonitor.DisableMonitoring function 575
IBSQLMonitor.EnableMonitoring 575
IBSQLMonitor.EnableMonitoring function 575
IBSQLMonitor.IIBSQLMonitorHook 567
IBSQLMonitor.IIBSQLMonitorHook.DBConnect 568
IBSQLMonitor.IIBSQLMonitorHook.DBDisconnect 569
IBSQLMonitor.IIBSQLMonitorHook.Enabled 574
IBSQLMonitor.IIBSQLMonitorHook.GetEnabled 569
IBSQLMonitor.IIBSQLMonitorHook.GetMonitorCount 569
IBSQLMonitor.IIBSQLMonitorHook.GetTraceFlags 569
IBSQLMonitor.IIBSQLMonitorHook.RegisterMonitor 570
IBSQLMonitor.IIBSQLMonitorHook.ReleaseMonitor 570
IBSQLMonitor.IIBSQLMonitorHook.SendMisc 570
IBSQLMonitor.IIBSQLMonitorHook.ServiceAttach 570
IBSQLMonitor.IIBSQLMonitorHook.ServiceDetach 570
IBSQLMonitor.IIBSQLMonitorHook.ServiceQuery 571
IBSQLMonitor.IIBSQLMonitorHook.ServiceStart 571
IBSQLMonitor.IIBSQLMonitorHook.SetEnabled 571
IBSQLMonitor.IIBSQLMonitorHook.SetTraceFlags 571
IBSQLMonitor.IIBSQLMonitorHook.SQLExecute 572
IBSQLMonitor.IIBSQLMonitorHook.SQLFetch 572
IBSQLMonitor.IIBSQLMonitorHook.SQLPrepare 572
IBSQLMonitor.IIBSQLMonitorHook.TraceFlags 574
IBSQLMonitor.IIBSQLMonitorHook.TRCommit 572
IBSQLMonitor.IIBSQLMonitorHook.TRCommitRetaining 572
IBSQLMonitor.IIBSQLMonitorHook.TRRollback 573
IBSQLMonitor.IIBSQLMonitorHook.TRRollbackRetaining 573
IBSQLMonitor.IIBSQLMonitorHook.TRStart 573
IBSQLMonitor.IIBSQLMonitorHook.UnregisterMonitor 573
IBSQLMonitor.MonitorHook 576
IBSQLMonitor.MonitorHook function 576

IBSQLMonitor.MonitoringEnabled 576
IBSQLMonitor.MonitoringEnabled function 576
IBSQLMonitor.TIBCustomSQLMonitor 561
IBSQLMonitor.TIBCustomSQLMonitor.Create 562
IBSQLMonitor.TIBCustomSQLMonitor.Destroy 562
IBSQLMonitor.TIBCustomSQLMonitor.Enabled 563
IBSQLMonitor.TIBCustomSQLMonitor.Handle 563
IBSQLMonitor.TIBCustomSQLMonitor.OnSQL 564
IBSQLMonitor.TIBCustomSQLMonitor.Release 562
IBSQLMonitor.TIBCustomSQLMonitor.TraceFlags 564
IBSQLMonitor.TIBSQLMonitor 564
IBSQLMonitor.TIBSQLMonitor.Enabled 565
IBSQLMonitor.TIBSQLMonitor.OnSQL 566
IBSQLMonitor.TIBSQLMonitor.TraceFlags 566
IBSQLMonitor.TSQLEvent 577
IBSQLMonitor.TSQLEvent type 577
IBSQLMonitor.WM_IBSQL_SQL_EVENT 578
IBSQLMonitor.WM_IBSQL_SQL_EVENT constant 578
IBSQLMonitor.WM_MAX_IBSQL_MONITOR 578
IBSQLMonitor.WM_MAX_IBSQL_MONITOR constant 578
IBSQLMonitor.WM_MIN_IBSQL_MONITOR 578
IBSQLMonitor.WM_MIN_IBSQL_MONITOR constant 578
IBStoredProc 581
IBStoredProc namespace 581
Classes 582
IBStoredProc.TIBStoredProc 582
IBStoredProc.TIBStoredProc.AfterDatabaseDisconnect 595
IBStoredProc.TIBStoredProc.AfterTransactionEnd 595
IBStoredProc.TIBStoredProc.BeforeDatabaseDisconnect 596
IBStoredProc.TIBStoredProc.BeforeTransactionEnd 596
IBStoredProc.TIBStoredProc.CopyParams 588
IBStoredProc.TIBStoredProc.Create 586
IBStoredProc.TIBStoredProc.DatabaseFree 596
IBStoredProc.TIBStoredProc.DefineProperties 588
IBStoredProc.TIBStoredProc.Destroy 587
IBStoredProc.TIBStoredProc.Disconnect 588
IBStoredProc.TIBStoredProc.ExecProc 588
IBStoredProc.TIBStoredProc.Filtered 592
IBStoredProc.TIBStoredProc.GetParamsCount 589
IBStoredProc.TIBStoredProc.InternalOpen 589
IBStoredProc.TIBStoredProc.OnFilterRecord 593
IBStoredProc.TIBStoredProc.ParamByName 589
IBStoredProc.TIBStoredProc.ParamCount 593
IBStoredProc.TIBStoredProc.Params 593
IBStoredProc.TIBStoredProc.Prepare 590
IBStoredProc.TIBStoredProc.Prepared 594
IBStoredProc.TIBStoredProc.PSExecute 590
IBStoredProc.TIBStoredProc.PSGetParams 590
IBStoredProc.TIBStoredProc.PSGetTableName 590
IBStoredProc.TIBStoredProc.PSSetCommandText 591
IBStoredProc.TIBStoredProc.PSSetParams 591
IBStoredProc.TIBStoredProc.SetFiltered 591
IBStoredProc.TIBStoredProc.SetPrepare 591
IBStoredProc.TIBStoredProc.SetPrepared 591
IBStoredProc.TIBStoredProc.SetProcName 592
IBStoredProc.TIBStoredProc.StmtHandle 594
IBStoredProc.TIBStoredProc.StoredProcedureNames 594
IBStoredProc.TIBStoredProc.StoredProcName 595
IBStoredProc.TIBStoredProc.TransactionFree 596
IBStoredProc.TIBStoredProc.UnPrepare 592
IBTable 599
IBTable namespace 599
Classes 600
Structs, Records, Enums 626
Types 627
IBTable.TIBTable 600
IBTable.TIBTable.Active 615
IBTable.TIBTable.AddIndex 606
IBTable.TIBTable.AfterDatabaseDisconnect 623
IBTable.TIBTable.AfterTransactionEnd 623
IBTable.TIBTable.BeforeDatabaseDisconnect 624
IBTable.TIBTable.BeforeTransactionEnd 624
IBTable.TIBTable.BufferChunks 615
IBTable.TIBTable.CachedUpdates 615
IBTable.TIBTable.Constraints 616
IBTable.TIBTable.Create 605
IBTable.TIBTable.CreateTable 606
IBTable.TIBTable.CurrentDBKey 616

IBTable.TIBTable.DatabaseFree 624
IBTable.TIBTable.DataEvent 607
IBTable.TIBTable.DefaultIndex 616
IBTable.TIBTable.DefChanged 607
IBTable.TIBTable.DeleteIndex 607
IBTable.TIBTable.DeleteTable 608
IBTable.TIBTable.Destroy 605
IBTable.TIBTable.DoOnNewRecord 608
IBTable.TIBTable.EmptyTable 608
IBTable.TIBTable.Exists 616
IBTable.TIBTable.FieldDefs 617
IBTable.TIBTable.Filter 617
IBTable.TIBTable.Filtered 617
IBTable.TIBTable.GetCanModify 608
IBTable.TIBTable.GetDataSource 609
IBTable.TIBTable.GetDetailLinkFields 609
IBTable.TIBTable.GetIndexNames 609
IBTable.TIBTable.GetIndexParams 609
IBTable.TIBTable.GotoCurrent 610
IBTable.TIBTable.IndexDefs 618
IBTable.TIBTable.IndexFieldCount 618
IBTable.TIBTable.IndexFieldNames 618
IBTable.TIBTable.IndexFields 619
IBTable.TIBTable.IndexName 619
IBTable.TIBTable.InitFieldDefs 610
IBTable.TIBTable.InternalClose 610
IBTable.TIBTable.InternalOpen 610
IBTable.TIBTable.InternalRefresh 611
IBTable.TIBTable.InternalRefreshRow 611
IBTable.TIBTable.LiveMode 619
IBTable.TIBTable.Locate 611
IBTable.TIBTable.MasterFields 620
IBTable.TIBTable.MasterSource 620
IBTable.TIBTable.OnFilterRecord 620
IBTable.TIBTable.PSGetDefaultOrder 612
IBTable.TIBTable.PSGetIndexDefs 612
IBTable.TIBTable.PSGetKeyFields 612
IBTable.TIBTable.PSGetTableName 612
IBTable.TIBTable.PSSetCommandText 613
IBTable.TIBTable.PSSetParams 613
IBTable.TIBTable.ReadOnly 621
IBTable.TIBTable.SetFiltered 613
IBTable.TIBTable.SetFilterOptions 613
IBTable.TIBTable.SetFilterText 614
IBTable.TIBTable.StoreDefs 621
IBTable.TIBTable.TableName 621
IBTable.TIBTable.TableNames 622
IBTable.TIBTable.TableTypes 622
IBTable.TIBTable.TransactionFree 624
IBTable.TIBTable.UniDirectional 622
IBTable.TIBTable.UpdateIndexDefs 614
IBTable.TIBTable.UpdateObject 623
IBTable.TIBTableType 626
IBTable.TIBTableType enumeration 626
IBTable.TIBTableTypes 627
IBTable.TIBTableTypes type 627
IBTable.TIndexName 627
IBTable.TIndexName type 627
IBUpdateSQL 629
IBUpdateSQL namespace 629
Classes 630
IBUpdateSQL.TIBUpdateSQL 630
IBUpdateSQL.TIBUpdateSQL.Apply 632
IBUpdateSQL.TIBUpdateSQL.Create 631
IBUpdateSQL.TIBUpdateSQL.DataSet 635
IBUpdateSQL.TIBUpdateSQL.DeleteSQL 635
IBUpdateSQL.TIBUpdateSQL.Destroy 631
IBUpdateSQL.TIBUpdateSQL.ExecSQL 633
IBUpdateSQL.TIBUpdateSQL.GetDataSet 633
IBUpdateSQL.TIBUpdateSQL.GetSQL 633
IBUpdateSQL.TIBUpdateSQL.InsertSQL 636
IBUpdateSQL.TIBUpdateSQL.ModifySQL 636
IBUpdateSQL.TIBUpdateSQL.Query 636
IBUpdateSQL.TIBUpdateSQL.SetDataSet 634
IBUpdateSQL.TIBUpdateSQL.SetParams 634
IBUpdateSQL.TIBUpdateSQL.SQL 637
IBUpdateSQL.TIBUpdateSQL.SQLChanged 634
IBUtils 639

IBUtils namespace 639
Constants 644
Functions 640
IBUtils.AddIBParamSQLForDetail 640
IBUtils.AddIBParamSQLForDetail function 640
IBUtils.CR 644
IBUtils.CR constant 644
IBUtils.CRLF 644
IBUtils.CRLF constant 644
IBUtils.ExtractIdentifier 640
IBUtils.ExtractIdentifier function 640
IBUtils.FormatIdentifier 641
IBUtils.FormatIdentifier function 641
IBUtils.FormatIdentifierValue 641
IBUtils.FormatIdentifierValue function 641
IBUtils.LF 645
IBUtils.LF constant 645
IBUtils.Max 641
IBUtils.Max function 641
IBUtils.Min 642
IBUtils.Min function 642
IBUtils.NULL_TERMINATOR 645
IBUtils.NULL_TERMINATOR constant 645
IBUtils.QuoteIdentifier 642
IBUtils.QuoteIdentifier function 642
IBUtils.RandomInteger 642
IBUtils.RandomInteger function 642
IBUtils.RandomString 643
IBUtils.RandomString function 643
IBUtils.StripString 643
IBUtils.StripString function 643
IBUtils.TAB 645
IBUtils.TAB constant 645
ibxeBlobCannotBeRead enumeration member 11
ibxeBlobCannotBeWritten enumeration member 11
ibxeBOF enumeration member 11
ibxeBufferNotSet enumeration member 11
ibxeCannotCreatePrimaryIndex enumeration member 11
ibxeCannotCreateSharedResource enumeration member 11
ibxeCannotDelete enumeration member 11
ibxeCannotDropSystemIndex enumeration member 11
ibxeCannotInsert enumeration member 11
ibxeCannotOpenNonSQLSelect enumeration member 11
ibxeCannotPost enumeration member 11
ibxeCannotRefresh enumeration member 11
ibxeCannotSetDatabase enumeration member 11
ibxeCannotSetTransaction enumeration member 11
ibxeCannotUpdate enumeration member 11
ibxeCantEndSharedTransaction enumeration member 11
ibxeCantPrintValue enumeration member 11
ibxeCircularDataLink enumeration member 11
ibxeCircularReference enumeration member 11
ibxeColumnIsNotNullable enumeration member 11
ibxeColumnListsDontMatch enumeration member 11
ibxeColumnTypesDontMatch enumeration member 11
ibxeConnectionTimeout enumeration member 11
ibxeDatabaseClosed enumeration member 11
ibxeDatabaseNameMissing enumeration member 11
ibxeDatabaseNotAssigned enumeration member 11
ibxeDatabaseOpen enumeration member 11
ibxeDatasetClosed enumeration member 11
ibxeDatasetOpen enumeration member 11
ibxeDataSetUniDirectional enumeration member 11
ibxeDelphiException enumeration member 11
ibxeDPBConstantNotSupported enumeration member 11
ibxeDPBConstantUnknown enumeration member 11
ibxeDPBConstantUnknownEx enumeration member 11
ibxeEmptyQuery enumeration member 11
ibxeEmptySQLStatement enumeration member 11
ibxeEOF enumeration member 11
ibxeEOFInComment enumeration member 11
ibxeEOFInString enumeration member 11
ibxeEOFReached enumeration member 11
ibxeEventAlreadyRegistered enumeration member 11
ibxeFieldNotFound enumeration member 11
ibxeFieldReadOnly enumeration member 11
ibxeFieldSizeMismatch enumeration member 11
ibxeFieldUnsupportedType enumeration member 11

ibxeFileAccessError enumeration member 11
ibxeIB60feature enumeration member 11
ibxeIndexFieldMissing enumeration member 11
ibxeInterBaseInstallMissing enumeration member 11
ibxeInterBaseMissing enumeration member 11
ibxeInTransaction enumeration member 11
ibxeInvalidBatchMove enumeration member 11
ibxeInvalidCancellation enumeration member 11
ibxeInvalidColumnIndex enumeration member 11
ibxeInvalidDataConversion enumeration member 11
ibxeInvalidEvent enumeration member 11
ibxeInvalidOnErrorResult enumeration member 11
ibxeInvalidOnStatusResult enumeration member 11
ibxeInvalidOption enumeration member 11
ibxeInvalidParamColumnIndex enumeration member 11
ibxeInvalidQueueing enumeration member 11
ibxeInvalidRegistration enumeration member 11
ibxeInvalidStatementHandle enumeration member 11
ibxeIsAExecuteProcedure enumeration member 11
ibxeIsASelectStatement enumeration member 11
ibxeMaximumEvents enumeration member 11
ibxeNoDatabasesInTransaction enumeration member 11
ibxeNoDestinationDirectory enumeration member 11
ibxeNoEventsRegistered enumeration member 11
ibxeNoFieldAccess enumeration member 11
ibxeNoOptionsSet enumeration member 11
ibxeNoProvider enumeration member 11
ibxeNoRecordsAffected enumeration member 11
ibxeNosourceDirectory enumeration member 11
ibxeNoStoredProcName enumeration member 11
ibxeNoTableName enumeration member 11
ibxeNotCachedUpdates enumeration member 11
ibxeNotEditing enumeration member 11
ibxeNotInTransaction enumeration member 11
ibxeNotLiveRequest enumeration member 11
ibxeNotPermitted enumeration member 11
ibxeNotSupported enumeration member 11
ibxeNoUninstallFile enumeration member 11
ibxeOperationCancelled enumeration member 11
ibxeOptionNeedsClient enumeration member 11
ibxeOptionNeedsServer enumeration member 11
ibxeOutputParsingError enumeration member 11
ibxeParamNameExpected enumeration member 11
ibxeQueryParamsError enumeration member 11
ibxeRequiredParamNotSet enumeration member 11
ibxeServerNameMissing enumeration member 11
ibxeServiceActive enumeration member 11
ibxeServiceInActive enumeration member 11
ibxeSPBConstantNotSupported enumeration member 11
ibxeSPBConstantUnknown enumeration member 11
ibxeSQLClosed enumeration member 11
ibxeSQLDialectInvalid enumeration member 11
ibxeSQLMonitorAlreadyPresent enumeration member 11
ibxeSQLOpen enumeration member 11
ibxeSQLParseError enumeration member 11
ibxeStartParamsError enumeration member 11
ibxeStringTooLarge enumeration member 11
ibxeSuccess enumeration member 11
ibxeTableNameMismatch enumeration member 11
ibxeTimeoutNegative enumeration member 11
ibxeTPBConstantNotSupported enumeration member 11
ibxeTPBConstantUnknown enumeration member 11
ibxeTPBConstantUnknownEx enumeration member 11
ibxeTransactionNotAssigned enumeration member 11
ibxeUnknownError enumeration member 11
ibxeUnknownPlan enumeration member 11
ibxeUnknownSQLDataType enumeration member 11
ibxeUpdateFailed enumeration member 11
ibxeUpdateWrongDB enumeration member 11
ibxeUpdateWrongTR enumeration member 11
ibxeUserAbort enumeration member 11
ibxeUseSpecificProcedures enumeration member 11
ibxeWindowsAPIError enumeration member 11
ibxeXSQLDAIndexOutOfRange enumeration member 11
ibxeXSQLDANameDoesNotExist enumeration member 11
Icon
    TApplicationMenu class 3354
Iconic

TOleContainer class 2566
IconSize
TApplicationMenu class 3354
ICorbaObj interface 2795
about ICorbaObj interface 2795
CorbaObject 2796
ICorbaObj Members 2795
ICorbaObj methods 2795
IsLocal 2796
Id
ITypeCode interface 2818
TLicenseInfo class 467
ID
TIBLicensingService class 436
TLimboTransactionInfo class 471
IDataSetComponent interface 2200
about IDataSetComponent interface 2200
DataSet 2202
GetDataSet 2201
GetXMLBroker 2201
GetXMLDataSet 2202
IDataSetComponent Members 2201
IDataSetComponent methods 2201
IDataSetComponent properties 2202
XMLBroker 2203
XMLDataSet 2203
IDataSetField interface 2204
about IDataSetField interface 2204
FieldName 2205
GetFieldName 2204
GetRowSetFieldAttributes 2205
IDataSetField Members 2204
IDataSetField methods 2204
IDataSetField properties 2205
SetFieldName 2205
IDataSetFields interface 2206
about IDataSetFields interface 2206
GetFieldsList 2207
GetVisibleFields 2207
HasStatusField 2207
IDataSetFields Members 2206
IDataSetFields methods 2206
IDataSetFields properties 2208
IsFieldInUse 2207
ValidateDataSet 2207
VisibleFields 2208
Identifier
TWebStatus class 2136
Idle
TCustomRibbon class 3382
IdleTimer
TIBDataBase class 191
TIBTransaction class 217
ieDelphiException enumeration member 330
ieInvalidOnErrorResult enumeration member 330
ieInvalidOnStatusResult enumeration member 330
ieInvalidOption enumeration member 330
ieNoDestinationDirectory enumeration member 330
ieNoOptionsSet enumeration member 330
ieNosourceDirectory enumeration member 330
ieNoUninstallFile enumeration member 330
IEnumNames interface 2417
about IEnumNames interface 2417
Clone 2418
IEnumNames Members 2417
IEnumNames methods 2418
Next 2418
Reset 2418
Skip 2418
ieOptionNeedsClient enumeration member 330
ieOptionNeedsServer enumeration member 330
ieSuccess enumeration member 330
IFormatColumn interface 2208
about IFormatColumn interface 2208
FormatColumnData 2209
FormatColumnHeading 2209
IFormatColumn Members 2208
IFormatColumn methods 2209

IGetContextProperties interface 2419
about IGetContextProperties interface 2419
Count 2419
EnumNames 2420
GetProperty 2420
IGetContextProperties Members 2419
IGetContextProperties methods 2419
IgnoreChecksum enumeration member 485
IgnoreChecksums enumeration member 477
IgnoreLimbo enumeration member 477
IHeadersSetter interface 919
about IHeadersSetter interface 919
IHeadersSetter Members 919
IHeadersSetter methods 919
SetHeadersInOut 920
IHTMLField interface 2209
about IHTMLField interface 2209
GetHTMLControlName 2210
HTMLControlName 2211
IHTMLField Members 2210
IHTMLField methods 2210
IHTMLField properties 2210
IHTMLForm interface 2211
about IHTMLForm interface 2211
GetHTMLFormName 2212
GetHTMLFormVarName 2212
HTMLFormName 2213
HTMLFormVarName 2213
IHTMLForm Members 2211
IHTMLForm methods 2212
IHTMLForm properties 2213
IIBEventNotifier interface 221
about IIBEventNotifier interface 221
GetAutoRegister 222
IIBEventNotifier Members 221
IIBEventNotifier methods 222
RegisterEvents 222
UnRegisterEvents 222
IIBSQLMonitorHook interface 567
about IIBSQLMonitorHook interface 567
DBConnect 568
DBDisconnect 569
Enabled 574
GetEnabled 569
GetMonitorCount 569
GetTraceFlags 569
IIBSQLMonitorHook Members 567
IIBSQLMonitorHook methods 568
IIBSQLMonitorHook properties 573
RegisterMonitor 570
ReleaseMonitor 570
SendMisc 570
ServiceAttach 570
ServiceDetach 570
ServiceQuery 571
ServiceStart 571
SetEnabled 571
SetTraceFlags 571
SQLExecute 572
SQLFetch 572
SQLPrepare 572
TraceFlags 574
TRCommit 572
TRCommitRetaining 572
TRRollback 573
TRRollbackRetaining 573
TRStart 573
UnregisterMonitor 573
IInetXWebDataSet interface 2213
about IInetXWebDataSet interface 2213
AddOnDataChange 2215
FetchParams 2215
FieldCount 2215
Fields 2216
IInetXWebDataSet Members 2214
IInetXWebDataSet methods 2214
ParamCount 2216
Params 2216

RemoveOnDataChange 2217
imAdam7 enumeration member 3021
ImageIndex
TListControlItem class 1047
TMenuItem class 1411
Images
TCustomListAction class 1025
TMainMenu class 1329
TMenu class 1345
TPopupMenu class 1439
TStaticListAction class 1058
TVirtualListAction class 1070
ImageType
TCustomImageList class 682
IMarshalInBuffer interface 2796
about IMarshalInBuffer interface 2796
Buffer 2797
GetAny 2798
GetChar 2798
GetDouble 2798
GetFloat 2798
GetLong 2798
GetObject 2799
GetShort 2799
GetText 2799
GetUnsignedChar 2799
GetUnsignedLong 2800
GetUnsignedShort 2800
GetWidechar 2800
GetWideText 2800
IMarshalInBuffer Members 2796
IMarshalInBuffer methods 2797
IMarshalOutBuffer interface 2800
about IMarshalOutBuffer interface 2800
Buffer 2802
IMarshalOutBuffer Members 2801
IMarshalOutBuffer methods 2801
PutAny 2802
PutChar 2802
PutDouble 2802
PutFloat 2802
PutLong 2803
PutObject 2803
PutShort 2803
PutText 2803
PutUnsignedChar 2804
PutUnsignedLong 2804
PutUnsignedShort 2804
PutWidechar 2804
PutWideText 2804
ImeMode
TMaskEdit class 1124
TRibbonComboBox class 3634
ImeName
TMaskEdit class 1124
TRibbonComboBox class 3634
IMethodHandler interface 2451
about IMethodHandler interface 2451
Execute 2452
IMethodHandler Members 2451
IMethodHandler methods 2451
InstanceToVariant 2452
ImgList 647
ImgList namespace 647
Classes 648
Structs, Records, Enums 685
Types 689
ImgList.TChangeLink 648
ImgList.TChangeLink.Change 649
ImgList.TChangeLink.Destroy 649
ImgList.TChangeLink.OnChange 650
ImgList.TChangeLink.Sender 650
ImgList.TColorDepth 685
ImgList.TColorDepth enumeration 685
ImgList.TCustomImageList 651
ImgList.TCustomImageList.Add 655
ImgList.TCustomImageList.AddIcon 655
ImgList.TCustomImageList.AddImage 656

ImgList.TCustomImageList.AddImages 656
ImgList.TCustomImageList.AddMasked 657
ImgList.TCustomImageList.AllocBy 675
ImgList.TCustomImageList.Assign 657
ImgList.TCustomImageList.AssignTo 658
ImgList.TCustomImageList.BkColor 676
ImgList.TCustomImageList.BlendColor 676
ImgList.TCustomImageList.Change 658
ImgList.TCustomImageList.Clear 658
ImgList.TCustomImageList.ColorDepth 677
ImgList.TCustomImageList.Count 677
ImgList.TCustomImageList.Create 653
ImgList.TCustomImageList.CreateSize 653
ImgList.TCustomImageList.DefineProperties 659
ImgList.TCustomImageList.Delete 659
ImgList.TCustomImageList.Destroy 654
ImgList.TCustomImageList.DoDraw 659
ImgList.TCustomImageList.Draw 660
ImgList.TCustomImageList.DrawingStyle 681
ImgList.TCustomImageList.DrawOverlay 663
ImgList.TCustomImageList.FileLoad 664
ImgList.TCustomImageList.GetBitmap 664
ImgList.TCustomImageList.GetHotSpot 665
ImgList.TCustomImageList.GetIcon 665
ImgList.TCustomImageList.GetImageBitmap 666
ImgList.TCustomImageList.GetImages 666
ImgList.TCustomImageList.GetInstRes 667
ImgList.TCustomImageList.GetMaskBitmap 668
ImgList.TCustomImageList.GetResource 668
ImgList.TCustomImageList.Handle 682
ImgList.TCustomImageList.HandleAllocated 669
ImgList.TCustomImageList.HandleNeeded 669
ImgList.TCustomImageList.Height 682
ImgList.TCustomImageList.ImageType 682
ImgList.TCustomImageList.Initialize 669
ImgList.TCustomImageList.Insert 670
ImgList.TCustomImageList.InsertIcon 670
ImgList.TCustomImageList.InsertMasked 670
ImgList.TCustomImageList.Masked 683
ImgList.TCustomImageList.Move 671
ImgList.TCustomImageList.OnChange 684
ImgList.TCustomImageList.Overlay 671
ImgList.TCustomImageList.ReadData 672
ImgList.TCustomImageList.RegisterChanges 672
ImgList.TCustomImageList.Replace 672
ImgList.TCustomImageList.ReplaceIcon 673
ImgList.TCustomImageList.ReplaceMasked 673
ImgList.TCustomImageList.ResInstLoad 673
ImgList.TCustomImageList.ResourceLoad 674
ImgList.TCustomImageList.SetSize 674
ImgList.TCustomImageList.ShareImages 683
ImgList.TCustomImageList.UnRegisterChanges 674
ImgList.TCustomImageList.Width 684
ImgList.TCustomImageList.WriteData 675
ImgList.TDrawingStyle 686
ImgList.TDrawingStyle enumeration 686
ImgList.TImageIndex 689
ImgList.TImageIndex type 689
ImgList.TImageListHandle 689
ImgList.TImageListHandle type 689
ImgList.TImageType 687
ImgList.TImageType enumeration 687
ImgList.TLoadResource 687
ImgList.TLoadResource enumeration 687
ImgList.TLoadResources 689
ImgList.TLoadResources type 689
ImgList.TOverlay 690
ImgList.TOverlay type 690
ImgList.TResType 688
ImgList.TResType enumeration 688
imNone enumeration member 3021
IMouse 691
IMouse namespace 691
Classes 693
Constants 711
Functions 706
Structs, Records, Enums 707
Types 709

IMouse.crPanAll 711
IMouse.crPanAll constant 711
IMouse.crPanDown 711
IMouse.crPanDown constant 711
IMouse.crPanDownLeft 712
IMouse.crPanDownLeft constant 712
IMouse.crPanDownRight 712
IMouse.crPanDownRight constant 712
IMouse.crPanLeft 712
IMouse.crPanLeft constant 712
IMouse.crPanLeftRight 713
IMouse.crPanLeftRight constant 713
IMouse.crPanRight 713
IMouse.crPanRight constant 713
IMouse.crPanUp 713
IMouse.crPanUp constant 713
IMouse.crPanUpDown 714
IMouse.crPanUpDown constant 714
IMouse.crPanUpLeft 714
IMouse.crPanUpLeft constant 714
IMouse.crPanUpRight 714
IMouse.crPanUpRight constant 714
IMouse.StartPan 706
IMouse.StartPan function 706
IMouse.TPanDirection 707
IMouse.TPanDirection enumeration 707
IMouse.TPanDirections 709
IMouse.TPanDirections type 709
IMouse.TPanningWindow 693
IMouse.TPanningWindow.Create 694
IMouse.TPanningWindow.CreateParams 695
IMouse.TPanningWindow.Destroy 694
IMouse.TPanningWindow.DoUpdate 696
IMouse.TPanningWindow.GetIsPanning 696
IMouse.TPanningWindow.GetPanOptions 696
IMouse.TPanningWindow.IsPanning 703
IMouse.TPanningWindow.MouseMove 697
IMouse.TPanningWindow.OnUpdate 704
IMouse.TPanningWindow.Paint 697
IMouse.TPanningWindow.PanControl 703
IMouse.TPanningWindow.PanInterval 704
IMouse.TPanningWindow.PanOptions 704
IMouse.TPanningWindow.SetPanCursor 698
IMouse.TPanningWindow.StartPanning 698
IMouse.TPanningWindow.StopPanning 698
IMouse.TPanningWindow.UpdateScroll 698
IMouse.TPanningWindow.WndProc 699
IMouse.TPanOption 707
IMouse.TPanOption enumeration 707
IMouse.TPanOptions 709
IMouse.TPanOptions type 709
IMouse.TUpdateEvent 709
IMouse.TUpdateEvent type 709
ImplAddElements
TApplyUpdatesButton class 1800
TCustomDataGrid class 1815
TCustomQueryButtons class 1838
TCustomQueryForm class 1844
TDataSetButton class 1863
TFieldGroup class 1872
TWebControlGroup class 2069
TWebForm class 2108
TWebListInput class 2118
TXMLDisplayGroup class 2183
TXMLDisplayReferenceGroup class 2198
ImplCanAddClass
TBaseInetXPageProducer class 2269
TCustomDataForm class 1810
TCustomDataGrid class 1815
TCustomDataNavigator class 1821
TCustomLayoutGroup class 1833
TCustomQueryButtons class 1838
TCustomQueryForm class 1845
TFieldGroup class 1872
TQueryFieldGroup class 1969
TWebControlGroup class 2070
TWebForm class 2109
ImplContent

TApplyUpdatesButton class 1800
TCustomDataForm class 1810
TCustomDataGrid class 1815
TCustomDataNavigator class 1821
TCustomFieldGroup class 1828
TCustomLayoutGroup class 1833
TCustomQueryButtons class 1838
TCustomQueryForm class 1845
TDataSetButton class 1863
TQueryButton class 1958
TWebButton class 2061
TWebControlGroup class 2070
TWebDataDisplay class 2080
TWebForm class 2109
ImplGetFieldName
TWebDataInput class 2091
ImplGetHTMLControlName
TWebDataDisplay class 2081
TWebDataInput class 2091
TWebStatus class 2136
ImplGetRowSetFieldAttributes
TWebDataInput class 2091
TWebRadioGroupInput class 2126
TWebTextAreaInput class 2143
TWebTextInput class 2149
ImplGetSubComponents
TCustomDataNavigator class 1822
TWebControlGroup class 2071
TXMLDisplayGroup class 2183
ImplIsMultipleRecordView
TCustomDataGrid class 1816
TXMLDisplayGroup class 2183
ImplIsReady
IBOA interface 2794
ImplLayoutButton
TFormLayout class 1918
ImplLayoutField
TFormLayout class 1919
ImplLayoutLabelAndField
TFormLayout class 1919
ImplLayoutTable
TFormLayout class 1920
ImplRestoreDefaults
TWebDataInput class 2092
InactiveTabFontColor
TCustomRibbonColorMap class 3404
TRibbonLunaColorMap class 3530
TRibbonObsidianColorMap class 3535
TRibbonSilverColorMap class 3554
IncludePathURL
TBaseInetXPageProducer class 2271
TInetXPageProducer class 2290
IncludesContent
TInetXPageElements class 2284
Increment
TRibbonSpinEdit class 3853
IncrementBy
TIBGeneratorField class 146
Index
TCustomRibbonPage class 3438
TIBXSQLVAR class 553
TKeyTipMatchItem class 3600
TOutlineNode class 2953
TRibbonPage class 3542
IndexDefs
TIBTable class 618
IndexFieldCount
TIBTable class 618
IndexFieldNames
TIBTable class 618
IndexFields
TIBTable class 619
IndexName
TIBTable class 619
IndexOf
THashedStringList class 751
THeaderList class 827
TMenuItem class 1381

IndexOfCaption
TRibbonTabs class 3562
IndexOfDBConst
TIBDataBase class 186
IndexOfName
THashedStringList class 752
IndexPages enumeration member 482
InfoToURI
TRemotableTypeRegistry class 882
IniFiles 715
IniFiles namespace 715
Classes 716
Structs, Records, Enums 787
Types 788
IniFiles.EIniFileException 716
IniFiles.PHashItem 788
IniFiles.PHashItem type 788
IniFiles.PPHashItem 788
IniFiles.PPHashItem type 788
IniFiles.TCustomIniFile 716
IniFiles.TCustomIniFile.Create 720
IniFiles.TCustomIniFile.DeleteKey 721
IniFiles.TCustomIniFile.EraseSection 722
IniFiles.TCustomIniFile.FileName 749
IniFiles.TCustomIniFile.ReadBinaryStream 722
IniFiles.TCustomIniFile.ReadBool 723
IniFiles.TCustomIniFile.ReadDate 726
IniFiles.TCustomIniFile.ReadDateTime 726
IniFiles.TCustomIniFile.ReadFloat 727
IniFiles.TCustomIniFile.ReadInteger 727
IniFiles.TCustomIniFile.ReadSection 730
IniFiles.TCustomIniFile.ReadSections 731
IniFiles.TCustomIniFile.ReadSectionValues 732
IniFiles.TCustomIniFile.ReadString 732
IniFiles.TCustomIniFile.ReadTime 735
IniFiles.TCustomIniFile.SectionExists 736
IniFiles.TCustomIniFile.UpdateFile 736
IniFiles.TCustomIniFile.ValueExists 736
IniFiles.TCustomIniFile.WriteBinaryStream 737
IniFiles.TCustomIniFile.WriteBool 737
IniFiles.TCustomIniFile.WriteDate 741
IniFiles.TCustomIniFile.WriteDateTime 741
IniFiles.TCustomIniFile.WriteFloat 742
IniFiles.TCustomIniFile.WriteInteger 742
IniFiles.TCustomIniFile.WriteString 745
IniFiles.TCustomIniFile.WriteTime 748
IniFiles.THashedStringList 749
IniFiles.THashedStringList.Changed 751
IniFiles.THashedStringList.Destroy 750
IniFiles.THashedStringList.IndexOf 751
IniFiles.THashedStringList.IndexOfName 752
IniFiles.THashItem 787
IniFiles.THashItem record 787
IniFiles.TIniFile 752
IniFiles.TIniFile.DeleteKey 757
IniFiles.TIniFile.Destroy 756
IniFiles.TIniFile.EraseSection 757
IniFiles.TIniFile.ReadSection 758
IniFiles.TIniFile.ReadSections 758
IniFiles.TIniFile.ReadSectionValues 759
IniFiles.TIniFile.ReadString 759
IniFiles.TIniFile.UpdateFile 762
IniFiles.TIniFile.WriteString 762
IniFiles.TMemIniFile 765
IniFiles.TMemIniFile.CaseSensitive 782
IniFiles.TMemIniFile.Clear 771
IniFiles.TMemIniFile.Create 770
IniFiles.TMemIniFile.DeleteKey 771
IniFiles.TMemIniFile.Destroy 770
IniFiles.TMemIniFile.Encoding 782
IniFiles.TMemIniFile.EraseSection 772
IniFiles.TMemIniFile.GetStrings 772
IniFiles.TMemIniFile.ReadSection 773
IniFiles.TMemIniFile.ReadSections 773
IniFiles.TMemIniFile.ReadSectionValues 774
IniFiles.TMemIniFile.ReadString 774
IniFiles.TMemIniFile.Rename 777
IniFiles.TMemIniFile.SetStrings 778

IniFiles.TMemIniFile.UpdateFile 778
IniFiles.TMemIniFile.WriteString 779
IniFiles.TStringHash 783
IniFiles.TStringHash.Add 784
IniFiles.TStringHash.Clear 784
IniFiles.TStringHash.Create 783
IniFiles.TStringHash.Destroy 784
IniFiles.TStringHash.Find 785
IniFiles.TStringHash.HashOf 785
IniFiles.TStringHash.Modify 785
IniFiles.TStringHash.Remove 785
IniFiles.TStringHash.ValueOf 786
Init
TSOAPAttachment class 899
InitAltRecBuffers
TPacketDataSet class 3192
InitControlData
TOleControl class 2668
InitControlInterface
TOleControl class 2668
InitData
TUpdateTree class 3208
InitDelta
TUpdateTree class 3208
InitErrorPacket
TUpdateTree class 3209
InitFieldDefs
TIBQuery class 384
TIBTable class 610
Initialize
TApplicationMenu class 3353
TCustomImageList class 669
TIBXSQLDA class 542
InitializeConflictBuffer
TCustomResolver class 3131
TDataSetResolver class 3181
TSQLResolver class 3203
InitiateAction
TMenuItem class 1382
InitKeyFields
TCustomResolver class 3132
InitObjectFromSOAP
IObjConverter interface 922
TSOAPDomConv class 2772
InitPutProcs
TDataPacketWriter class 3140
InitRecord
TIBCustomDataSet class 78
InitResponse
TISAPIResponse class 981
InitServerData
TOleServer class 2715
InitTreeData
TCustomResolver class 3132
TSQLResolver class 3203
InputType
TQueryButton class 1956
Insert
TCustomImageList class 670
TCustomOutline class 2853
TMenuItem class 1382
TOptionItems class 3939
TPNGPointerList class 3001
InsertCount
TIBDatabaseInfo class 251
InsertIcon
TCustomImageList class 670
InsertMasked
TCustomImageList class 670
InsertMenus
TOleControl class 2668
InsertNewLine
TMenuItem class 1383
InsertNewLineAfter
TMenuItem class 1383
InsertNewLineBefore
TMenuItem class 1384
InsertNode

TOutlineNode class 2949
InsertObject
TCustomOutline class 2853
InsertObjectDialog
TOleContainer class 2543
InsertSQL
TIBCustomDataSet class 100
TIBDataSet class 128
TIBUpdateSQL class 636
InSplit
TRibbonLargeSplitButton class 3749
TRibbonSmallSplitButton class 3794
InstallCheck
TIBInstall class 314
InstallerError
EIBInstall class 308
InstallError
EIBInstall class 309
InstallExecute
TIBInstall class 314
InstallOptions
TIBInstall class 315
Instance
TObjectDispatch class 2488
InstanceToVariant
IMethodHandler interface 2452
InterfaceSupportsErrorInfo
TLocalAppServer class 3187
InternalAddRecord
TIBCustomDataSet class 78
InternalApplyUpdates
TBaseProvider class 3084
TCustomProvider class 3110
TDataSetProvider class 3154
InternalBatchInput
TIBCustomDataSet class 79
InternalBatchOutput
TIBCustomDataSet class 79
InternalBeforeResolve
TCustomResolver class 3133
TDataSetResolver class 3181
InternalCancel
TIBCustomDataSet class 79
InternalClose
TIBCustomDataSet class 79
TIBTable class 610
TPacketDataSet class 3193
InternalDelete
TIBCustomDataSet class 79
InternalDoUpdate
TSQLResolver class 3204
InternalExecQuery
TIBCustomDataSet class 80
InternalExecute
TCustomProvider class 3111
TDataSetProvider class 3154
InternalFirst
TIBCustomDataSet class 80
InternalGetParams
TCustomProvider class 3111
TDataSetProvider class 3155
InternalGetRecords
TBaseProvider class 3085
TCustomProvider class 3112
TDataSetProvider class 3155
InternalGotoBookmark
TIBCustomDataSet class 80
InternalHandleException
TIBCustomDataSet class 80
InternalInitFieldDefs
TIBCustomDataSet class 81
InternalInitRecord
TIBCustomDataSet class 81
TPacketDataSet class 3193
InternalInsert
TIBCustomDataSet class 81
InternalLast
TIBCustomDataSet class 81

InternalOpen
TIBCustomDataSet class 81
TIBDataSet class 119
TIBQuery class 384
TIBStoredProc class 589
TIBTable class 610
TPacketDataSet class 3193
InternalPost
TIBCustomDataSet class 82
InternalPrepare
TIBCustomDataSet class 82
InternalPrepared
TIBCustomDataSet class 100
InternalRefresh
TIBCustomDataSet class 82
TIBTable class 611
InternalRefreshRow
TIBCustomDataSet class 82
TIBTable class 611
InternalRemove
TOutlineNode class 2949
InternalRowRequest
TBaseProvider class 3086
TCustomProvider class 3113
TDataSetProvider class 3156
InternalServiceQuery
TIBCustomService class 428
InternalServiceStart
TIBControlService class 423
InternalSetCacheFile
TSOAPAttachment class 900
InternalSetParamsFromCursor
TIBCustomDataSet class 83
InternalSetSourceStream
TSOAPAttachment class 900
InternalSetToRecord
TIBCustomDataSet class 83
InternalTransaction
TIBDataBase class 192
InternalUnPrepare
TIBCustomDataSet class 83
InternalUpdateRecord
TCustomResolver class 3133
Interval
TIBTimer class 206
IntfInfo 789
IntfInfo namespace 789
Classes 791
Constants 804
Functions 792
Structs, Records, Enums 797
Types 801
IntfInfo.CallingConventionName 804
IntfInfo.CallingConventionName constant 804
IntfInfo.EInterfaceRTTIException 791
IntfInfo.GetClsMemberTypeInfo 792
IntfInfo.GetClsMemberTypeInfo function 792
IntfInfo.GetDynArrayElTypeInfo 792
IntfInfo.GetDynArrayElTypeInfo function 792
IntfInfo.GetDynArrayNextInfo 793
IntfInfo.GetDynArrayNextInfo function 793
IntfInfo.GetDynArrayNextInfo2 793
IntfInfo.GetDynArrayNextInfo2 function 793
IntfInfo.GetIntfMetaData 793, 794
IntfInfo.GetIntfMetaData function 793, 794
IntfInfo.GetMethNum 794
IntfInfo.GetMethNum function 794
IntfInfo.GetPropListFlat 794
IntfInfo.GetPropListFlat function 794
IntfInfo.IsStoredPropConst 795
IntfInfo.IsStoredPropConst function 795
IntfInfo.OtherTypeName 795
IntfInfo.OtherTypeName function 795
IntfInfo.PIntfMetaData 801
IntfInfo.PIntfMetaData type 801
IntfInfo.PIntfMethEntry 801
IntfInfo.PIntfMethEntry type 801
IntfInfo.PIntfParamEntry 802

IntfInfo.PIntfParamEntry type 802
IntfInfo.SameTypeInfo 795
IntfInfo.SameTypeInfo function 795
IntfInfo.TCallConv 797
IntfInfo.TCallConv enumeration 797
IntfInfo.TFillMethodArrayOpt 797
IntfInfo.TFillMethodArrayOpt enumeration 797
IntfInfo.TIntfMetaData 798
IntfInfo.TIntfMetaData record 798
IntfInfo.TIntfMethEntry 799
IntfInfo.TIntfMethEntry record 799
IntfInfo.TIntfMethEntryArray 802
IntfInfo.TIntfMethEntryArray type 802
IntfInfo.TIntfParamEntry 800
IntfInfo.TIntfParamEntry record 800
IntfInfo.TIntfParamEntryArray 802
IntfInfo.TIntfParamEntryArray type 802
IntfInfo.TPIntfMethEntryArray 803
IntfInfo.TPIntfMethEntryArray type 803
IntfInfo.TypeInfoNames 804
IntfInfo.TypeInfoNames constant 804
IntfInfo.TypeNamesMatch 796
IntfInfo.TypeNamesMatch function 796
IntfMD
    TRIO class 4059
InTransaction
    TIBTransaction class 218
Invalidate
    TCustomRibbonPage class 3432
InvalidateCell
    TCustomOutline class 2854
InvalidateHeaderArea
    TCustomRibbon class 3382
InvalidateScrollButtons
    TCustomRibbon class 3382
InvalidateSystemMenuArea
    TCustomRibbon class 3383
InvalidateTabArea
    TCustomRibbon class 3383
InvContextToMsg
    IOPConvert interface 2733
    TOPToSoapDomConvert class 2752
Invoke
    IStub interface 2814
    TAutoDispatch class 2503
    TEventDispatch class 2637
    TLocalAppServer class 3187
    TObjectDispatch class 2488
    TOleControl class 2668
    TServerEventDispatch class 2723
InvokeEvent
    TOleControl class 2669
    TOleServer class 2715
InvokeMethod
    TOleControl class 2669
InvokeRegistry 807
InvokeRegistry namespace 807
    Classes 810
    Constants 956
    Functions 929
    Interfaces 919
    Structs, Records, Enums 934
    Types 946
    Variables 955
InvokeRegistry.AppNameSpacePrefix 955
InvokeRegistry.AppNameSpacePrefix variable 955
InvokeRegistry.AS_ATTRIBUTE 956
InvokeRegistry.AS_ATTRIBUTE constant 956
InvokeRegistry.eHeaderMethodType 934
InvokeRegistry.eHeaderMethodType enumeration 934
InvokeRegistry.ERemotableException 810
InvokeRegistry.ERemotableException.Create 811
InvokeRegistry.ERemotableException.FaultActor 812
InvokeRegistry.ERemotableException.FaultCode 812
InvokeRegistry.ERemotableException.FaultDetail 813
InvokeRegistry.ERemotableExceptionClass 946
InvokeRegistry.ERemotableExceptionClass type 946
InvokeRegistry.ETypeRegistryException 813

InvokeRegistry.ExtNameMapItem 935
InvokeRegistry.ExtNameMapItem record 935
InvokeRegistry.GetRemotableDataContext 929
InvokeRegistry.GetRemotableDataContext function 929
InvokeRegistry.IHeadersSetter 919
InvokeRegistry.IHeadersSetter.SetHeadersInOut 920
InvokeRegistry.IntfExceptionItem 935
InvokeRegistry.IntfExceptionItem record 935
InvokeRegistry.IntfHeaderItem 936
InvokeRegistry.IntfHeaderItem record 936
InvokeRegistry.InvRegClassEntry 937
InvokeRegistry.InvRegClassEntry record 937
InvokeRegistry.InvRegIntfEntry 937
InvokeRegistry.InvRegIntfEntry record 937
InvokeRegistry.InvRegistry 930
InvokeRegistry.InvRegistry function 930
InvokeRegistry.InvString 947
InvokeRegistry.InvString type 947
InvokeRegistry.InvStringDynArray 947
InvokeRegistry.InvStringDynArray type 947
InvokeRegistry.IObjConverter 920
InvokeRegistry.IObjConverter.AddAttachment 921
InvokeRegistry.IObjConverter.FindAttachment 922
InvokeRegistry.IObjConverter.InitObjectFromSOAP 922
InvokeRegistry.IObjConverter.ObjInstanceToSOAP 922
InvokeRegistry.ISOAPHeaders 923
InvokeRegistry.ISOAPHeaders.Get 925
InvokeRegistry.ISOAPHeaders.GetOwnsSentHeaders 925
InvokeRegistry.ISOAPHeaders.OwnsSentHeaders 927
InvokeRegistry.ISOAPHeaders.Send 926
InvokeRegistry.ISOAPHeaders.SendAt 926
InvokeRegistry.ISOAPHeaders.SendCount 927
InvokeRegistry.ISOAPHeaders.SetOwnsSentHeaders 927
InvokeRegistry.MethParamNameMapItem 939
InvokeRegistry.MethParamNameMapItem record 939
InvokeRegistry.ObjectConvertOptions 939
InvokeRegistry.ObjectConvertOptions enumeration 939
InvokeRegistry.PTRemotable 947
InvokeRegistry.PTRemotable type 947
InvokeRegistry.RemClassRegistry 930
InvokeRegistry.RemClassRegistry function 930
InvokeRegistry.RemTypeRegistry 931
InvokeRegistry.RemTypeRegistry function 931
InvokeRegistry.SerializationOptions 940
InvokeRegistry.SerializationOptions enumeration 940
InvokeRegistry.SetRemotableDataContext 932
InvokeRegistry.SetRemotableDataContext function 932
InvokeRegistry.SubstituteStrings 933
InvokeRegistry.SubstituteStrings function 933
InvokeRegistry.TCreateInstanceProc 948
InvokeRegistry.TCreateInstanceProc type 948
InvokeRegistry.TDataContext 814
InvokeRegistry.TDataContext.AddDynArrayToClear 818
InvokeRegistry.TDataContext.AddObjectToDestroy 819
InvokeRegistry.TDataContext.AddStrToClear 819
InvokeRegistry.TDataContext.AddUStrToClear 820
InvokeRegistry.TDataContext.AddVariantToClear 820
InvokeRegistry.TDataContext.AddWStrToClear 821
InvokeRegistry.TDataContext.AllocData 821
InvokeRegistry.TDataContext.Create 815
InvokeRegistry.TDataContext.Data 816
InvokeRegistry.TDataContext.DataOffset 816
InvokeRegistry.TDataContext.DataP 817
InvokeRegistry.TDataContext.Destroy 815
InvokeRegistry.TDataContext.DynArrayToClear 817
InvokeRegistry.TDataContext.FObjsToDestroy 817
InvokeRegistry.TDataContext.GetDataPointer 822
InvokeRegistry.TDataContext.RemoveObjectToDestroy 822
InvokeRegistry.TDataContext.SetDataPointer 822
InvokeRegistry.TDataContext.StrToClear 817
InvokeRegistry.TDataContext.UStrToClear 817
InvokeRegistry.TDataContext.VarToClear 818
InvokeRegistry.TDataContext.WStrToClear 818
InvokeRegistry.TDynToClear 941
InvokeRegistry.TDynToClear record 941
InvokeRegistry.TExceptionItemArray 948
InvokeRegistry.TExceptionItemArray type 948
InvokeRegistry.THeaderItemArray 948

InvokeRegistry.THeaderItemArray type 948
InvokeRegistry.THeaderList 823
InvokeRegistry.THeaderList.Add 825
InvokeRegistry.THeaderList.Clear 826
InvokeRegistry.THeaderList.Count 828
InvokeRegistry.THeaderList.Create 824
InvokeRegistry.THeaderList.Destroy 824
InvokeRegistry.THeaderList.Extract 826
InvokeRegistry.THeaderList.GetCount 826
InvokeRegistry.THeaderList.GetHeader 827
InvokeRegistry.THeaderList.GetOwnsObjects 827
InvokeRegistry.THeaderList.Headers 828
InvokeRegistry.THeaderList.IndexOf 827
InvokeRegistry.THeaderList.OwnsObjects 829
InvokeRegistry.THeaderList.SetOwnsObjects 827
InvokeRegistry.THeaderMethodTypeArray 949
InvokeRegistry.THeaderMethodTypeArray type 949
InvokeRegistry.TIntfInvokeOption 942
InvokeRegistry.TIntfInvokeOption enumeration 942
InvokeRegistry.TIntfInvokeOptions 949
InvokeRegistry.TIntfInvokeOptions type 949
InvokeRegistry.TInvContext 829
InvokeRegistry.TInvContext.AllocServerData 831
InvokeRegistry.TInvContext.GetParamPointer 831
InvokeRegistry.TInvContext.GetResultPointer 832
InvokeRegistry.TInvContext.SetMethodInfo 832
InvokeRegistry.TInvContext.SetParamPointer 833
InvokeRegistry.TInvContext.SetResultPointer 833
InvokeRegistry.TInvokableClass 833
InvokeRegistry.TInvokableClass._AddRef 836
InvokeRegistry.TInvokableClass._Release 837
InvokeRegistry.TInvokableClass.AfterConstruction 837
InvokeRegistry.TInvokableClass.BeforeDestruction 838
InvokeRegistry.TInvokableClass.Create 835
InvokeRegistry.TInvokableClass.FRefCount 836
InvokeRegistry.TInvokableClass.FSOAPHeaders 836
InvokeRegistry.TInvokableClass.NewInstance 838
InvokeRegistry.TInvokableClass.QueryInterface 839
InvokeRegistry.TInvokableClass.RefCount 839
InvokeRegistry.TInvokableClassClass 949
InvokeRegistry.TInvokableClassClass type 949
InvokeRegistry.TInvokableClassRegistry 840
InvokeRegistry.TInvokableClassRegistry.Create 842
InvokeRegistry.TInvokableClassRegistry.Destroy 842
InvokeRegistry.TInvokableClassRegistry.GetActionURIOfIID 844
InvokeRegistry.TInvokableClassRegistry.GetActionURIOfInfo 844
InvokeRegistry.TInvokableClassRegistry.GetClassFromIntfInfo 845
InvokeRegistry.TInvokableClassRegistry.GetExceptionInfoForInterface 845
InvokeRegistry.TInvokableClassRegistry.GetHeaderClass 846
InvokeRegistry.TInvokableClassRegistry.GetHeaderInfoForInterface 846
InvokeRegistry.TInvokableClassRegistry.GetHeaderName 847
InvokeRegistry.TInvokableClassRegistry.GetHeaderNamespace 848
InvokeRegistry.TInvokableClassRegistry.GetInfoForURI 848
InvokeRegistry.TInvokableClassRegistry.GetInterfaceCount 849
InvokeRegistry.TInvokableClassRegistry.GetInterfaceExternalName 849
InvokeRegistry.TInvokableClassRegistry.GetInterfaceInfoFromName 850
InvokeRegistry.TInvokableClassRegistry.GetInterfaceTypeInfo 850
InvokeRegistry.TInvokableClassRegistry.GetIntfInvokeOptions 851
InvokeRegistry.TInvokableClassRegistry.GetInvokableObjectFromClass 852
InvokeRegistry.TInvokableClassRegistry.GetMethExternalName 852
InvokeRegistry.TInvokableClassRegistry.GetMethInternalName 852
InvokeRegistry.TInvokableClassRegistry.GetNamespaceByGUID 853
InvokeRegistry.TInvokableClassRegistry.GetParamExternalName 853

InvokeRegistry.TInvokableClassRegistry.GetParamInternalName 854
InvokeRegistry.TInvokableClassRegistry.GetRegInterfaceEntry 854
InvokeRegistry.TInvokableClassRegistry.GetRequestHeaderInfoForInterface 855
InvokeRegistry.TInvokableClassRegistry.GetResponseHeaderInfoForInterface 855
InvokeRegistry.TInvokableClassRegistry.GetReturnParamNames 856
InvokeRegistry.TInvokableClassRegistry.GetUDDIInfo 856
InvokeRegistry.TInvokableClassRegistry.GetWSDLEncoding 857
InvokeRegistry.TInvokableClassRegistry.HasRegInterfaceImpl 858
InvokeRegistry.TInvokableClassRegistry.RegisterAllSOAPActions 859
InvokeRegistry.TInvokableClassRegistry.RegisterDefaultSOAPAction 859
InvokeRegistry.TInvokableClassRegistry.RegisterException 860
InvokeRegistry.TInvokableClassRegistry.RegisterExceptionMethod 860
InvokeRegistry.TInvokableClassRegistry.RegisterExternalMethName 861
InvokeRegistry.TInvokableClassRegistry.RegisterExternalParamName 861
InvokeRegistry.TInvokableClassRegistry.RegisterHeaderClass 862
InvokeRegistry.TInvokableClassRegistry.RegisterHeaderMethod 863
InvokeRegistry.TInvokableClassRegistry.RegisterInterface 864
InvokeRegistry.TInvokableClassRegistry.RegisterInvokableClass 865
InvokeRegistry.TInvokableClassRegistry.RegisterInvokeOptions 866
InvokeRegistry.TInvokableClassRegistry.RegisterReturnParamNames 867
InvokeRegistry.TInvokableClassRegistry.RegisterUDDIInfo 867
InvokeRegistry.TInvokableClassRegistry.UnRegisterInterface 867
InvokeRegistry.TInvokableClassRegistry.UnRegisterInvokableClass 868
InvokeRegistry.TInvRegIntfEntryArray 950
InvokeRegistry.TInvRegIntfEntryArray type 950
InvokeRegistry.TObjectConvertOptions 950
InvokeRegistry.TObjectConvertOptions type 950
InvokeRegistry.TObjMultiOptions 943
InvokeRegistry.TObjMultiOptions enumeration 943
InvokeRegistry.TPascalRemotableClassRegistry 950
InvokeRegistry.TPascalRemotableClassRegistry type 950
InvokeRegistry.TPascalRemotableTypeRegistry 951
InvokeRegistry.TPascalRemotableTypeRegistry type 951
InvokeRegistry.TRemHeaderEntry 944
InvokeRegistry.TRemHeaderEntry record 944
InvokeRegistry.TRemotable 868
InvokeRegistry.TRemotable.Create 870
InvokeRegistry.TRemotable.DataContext 873
InvokeRegistry.TRemotable.Destroy 870
InvokeRegistry.TRemotable.FSerializationOptions 871
InvokeRegistry.TRemotable.ObjectToSOAP 871
InvokeRegistry.TRemotable.SerializationOptions 873
InvokeRegistry.TRemotable.SOAPToObject 872
InvokeRegistry.TRemotableClass 951
InvokeRegistry.TRemotableClass type 951
InvokeRegistry.TRemotableClassRegistry 951
InvokeRegistry.TRemotableClassRegistry type 951
InvokeRegistry.TRemotableTypeRegistry 874
InvokeRegistry.TRemotableTypeRegistry.AutoRegisterNativeTypes 891
InvokeRegistry.TRemotableTypeRegistry.ClassOptions 877
InvokeRegistry.TRemotableTypeRegistry.ClassToURI 878
InvokeRegistry.TRemotableTypeRegistry.Create 876
InvokeRegistry.TRemotableTypeRegistry.Destroy 876
InvokeRegistry.TRemotableTypeRegistry.GetExternalPropName 879
InvokeRegistry.TRemotableTypeRegistry.GetInternalPropName 879

InvokeRegistry.TRemotableTypeRegistry.GetURICount 879
InvokeRegistry.TRemotableTypeRegistry.GetURIMap 880
InvokeRegistry.TRemotableTypeRegistry.GetVarTypeFromXSD 881
InvokeRegistry.TRemotableTypeRegistry.GetXSDInfoForClass 881
InvokeRegistry.TRemotableTypeRegistry.InfoToURI 882
InvokeRegistry.TRemotableTypeRegistry.IsClassScalar 882
InvokeRegistry.TRemotableTypeRegistry.Lock 883
InvokeRegistry.TRemotableTypeRegistry.RegisterExternalPropName 883
InvokeRegistry.TRemotableTypeRegistry.RegisterHolderClsMember 883
InvokeRegistry.TRemotableTypeRegistry.RegisterSerializeOptions 884
InvokeRegistry.TRemotableTypeRegistry.RegisterXSClass 885
InvokeRegistry.TRemotableTypeRegistry.RegisterXSInfo 886
InvokeRegistry.TRemotableTypeRegistry.SerializeOptions 887
InvokeRegistry.TRemotableTypeRegistry.TypeInfoToXSD 887
InvokeRegistry.TRemotableTypeRegistry.UnLock 888
InvokeRegistry.TRemotableTypeRegistry.UnRegisterXSClass 888
InvokeRegistry.TRemotableTypeRegistry.UnRegisterXSInfo 889
InvokeRegistry.TRemotableTypeRegistry.URIToClass 889
InvokeRegistry.TRemotableTypeRegistry.URIToInfo 890
InvokeRegistry.TRemotableTypeRegistry.VariantToInfo 890
InvokeRegistry.TRemotableTypeRegistry.XSDToTypeInfo 891
InvokeRegistry.TRemotableXS 892
InvokeRegistry.TRemotableXS.NativeToXS 893
InvokeRegistry.TRemotableXS.ObjectToSOAP 894
InvokeRegistry.TRemotableXS.SOAPToObject 894
InvokeRegistry.TRemotableXS.XSToNative 895
InvokeRegistry.TRemotableXSClass 952
InvokeRegistry.TRemotableXSClass type 952
InvokeRegistry.TRemRegEntry 944
InvokeRegistry.TRemRegEntry record 944
InvokeRegistry.TRemRegEntryArray 952
InvokeRegistry.TRemRegEntryArray type 952
InvokeRegistry.TRequiredArray 952
InvokeRegistry.TRequiredArray type 952
InvokeRegistry.TSerializationOptions 953
InvokeRegistry.TSerializationOptions type 953
InvokeRegistry.TSOAPAttachment 895
InvokeRegistry.TSOAPAttachment.CacheFile 904
InvokeRegistry.TSOAPAttachment.CacheFilePersist 905
InvokeRegistry.TSOAPAttachment.ContentType 905
InvokeRegistry.TSOAPAttachment.Create 897
InvokeRegistry.TSOAPAttachment.Destroy 898
InvokeRegistry.TSOAPAttachment.Encoding 906
InvokeRegistry.TSOAPAttachment.GetSourceStream 899
InvokeRegistry.TSOAPAttachment.Headers 906
InvokeRegistry.TSOAPAttachment.Init 899
InvokeRegistry.TSOAPAttachment.InternalSetCacheFile 900
InvokeRegistry.TSOAPAttachment.InternalSetSourceStream 900
InvokeRegistry.TSOAPAttachment.ObjectToSOAP 900
InvokeRegistry.TSOAPAttachment.Ownership 907
InvokeRegistry.TSOAPAttachment.SaveToFile 901
InvokeRegistry.TSOAPAttachment.SaveToStream 901
InvokeRegistry.TSOAPAttachment.SetOwnership 902
InvokeRegistry.TSOAPAttachment.SetSourceFile 902
InvokeRegistry.TSOAPAttachment.SetSourceStream 903
InvokeRegistry.TSOAPAttachment.SetSourceString 903
InvokeRegistry.TSOAPAttachment.SOAPToObject 903
InvokeRegistry.TSOAPAttachment.SourceStream 907
InvokeRegistry.TSOAPAttachment.SourceString 908
InvokeRegistry.TSOAPAttachmentClass 953
InvokeRegistry.TSOAPAttachmentClass type 953
InvokeRegistry.TSOAPHeader 908
InvokeRegistry.TSOAPHeader.Actor 911
InvokeRegistry.TSOAPHeader.MustUnderstand 912
InvokeRegistry.TSOAPHeader.ObjectToSOAP 910
InvokeRegistry.TSOAPHeader.SOAPToObject 911
InvokeRegistry.TSOAPHeaderClass 953
InvokeRegistry.TSOAPHeaderClass type 953
InvokeRegistry.TSOAPHeaders 912
InvokeRegistry.TSOAPHeaders.Get 914
InvokeRegistry.TSOAPHeaders.GetOwnsSentHeaders 914
InvokeRegistry.TSOAPHeaders.Send 915

InvokeRegistry.TSOAPHeaders.SendAt 915
InvokeRegistry.TSOAPHeaders.SendCount 916
InvokeRegistry.TSOAPHeaders.SetOwnsSentHeaders 916
InvokeRegistry.TSOAPHeadersBase 916
InvokeRegistry.TSOAPHeadersBase.FHeadersInbound 917
InvokeRegistry.TSOAPHeadersBase.FHeadersOutBound 918
InvokeRegistry.TSOAPHeadersBase.SetHeadersInOut 918
InvokeRegistry.XMLBase64Types 956
InvokeRegistry.XMLBase64Types constant 956
InvokeRegistry.XMLSchemaInstNamepspaces 957
InvokeRegistry.XMLSchemaInstNamepspaces constant 957
InvokeRegistry.XMLSchemaNamepspaces 957
InvokeRegistry.XMLSchemaNamepspaces constant 957
IObjConverter interface 920
about IObjConverter interface 920
AddAttachment 921
FindAttachment 922
InitObjectFromSOAP 922
IObjConverter Members 921
IObjConverter methods 921
ObjInstanceToSOAP 922
IObjectContext interface 2420
about IObjectContext interface 2420
CreateInstance 2421
DisableCommit 2421
EnableCommit 2422
IObjectContext Members 2420
IObjectContext methods 2421
IsCallerInRole 2422
IsInTransaction 2422
IsSecurityEnabled 2422
SetAbort 2422
SetComplete 2423
IObjectContextActivity interface 2423
about IObjectContextActivity interface 2423
GetActivityId 2424
IObjectContextActivity Members 2423
IObjectContextActivity methods 2423
IObjectControl interface 2424
about IObjectControl interface 2424
Activate 2425
CanBePooled 2425
Deactivate 2425
IObjectControl Members 2424
IObjectControl methods 2424
ioDefault enumeration member 942
ioDocument enumeration member 942
ioHasAllSOAPActions enumeration member 942
ioHasDefaultSOAPAction enumeration member 942
ioHasNamespace enumeration member 942
ioHasReturnParamNames enumeration member 942
ioHasUDDIInfo enumeration member 942
ioIsAppServerSOAP enumeration member 942
ioLiteral enumeration member 942
IOPConvert interface 2730
about IOPConvert interface 2730
Attachments 2738
Encoding 2738
GetAttachments 2732
GetEncoding 2732
GetOptions 2732
GetTempDir 2732
InvContextToMsg 2733
IOPConvert Members 2731
IOPConvert methods 2731
IOPConvert properties 2738
MakeFault 2733
MakeResponse 2734
MsgToInvContext 2735
MsgToInvContext method 2734
Options 2739
ProcessResponse 2736
ProcessResponse method 2735
SetAttachments 2736
SetEncoding 2737
SetOptions 2737
SetTempDir 2737
TempDir 2739

IORB interface 2805
about IORB interface 2805
BOAInit 2806
CreateAliasTC 2806
CreateArrayTC 2806
CreateObjRefTC 2807
CreateSequenceTC 2807
CreateStructTC 2807
CreateTC 2807
DispatchStruct 2807
FindRepositoryTC 2808
IORB Members 2805
IORB methods 2805
MakeAny 2808
ObjectToString 2808
Orb 2808
Shutdown 2809
StringToObject 2809
IProviderContainer interface 3213
about IProviderContainer interface 3213
IProviderContainer Members 3213
IProviderContainer methods 3213
RegisterProvider 3214
UnRegisterProvider 3214
IQueryField interface 2217
about IQueryField interface 2217
GetParamName 2218
GetText 2218
IQueryField Members 2217
IQueryField methods 2218
IQueryField properties 2219
ParamName 2219
SetParamName 2218
SetText 2219
Text 2219
IQueryFields interface 2220
about IQueryFields interface 2220
GetParamsList 2220
IQueryFields Members 2220
IQueryFields methods 2220
IsParamInUse 2221
ValidateDataSet 2221
IQueryForm interface 2221
about IQueryForm interface 2221
IRestoreDefaults interface 2221
about IRestoreDefaults interface 2221
IRestoreDefaults Members 2222
IRestoreDefaults methods 2222
RestoreDefaults 2222
IRibbon interface 3566
about IRibbon interface 3566
GetRibbon 3567
IRibbon Members 3566
IRibbon methods 3567
IRibbon properties 3567
Ribbon 3567
SetRibbon 3567
IRibbonBarKeyTip interface 3912
about IRibbonBarKeyTip interface 3912
BarHandleKeyTips 3913
BarHideKeyTips 3913
BarShowKeyTips 3914
IRibbonBarKeyTip Members 3912
IRibbonBarKeyTip methods 3913
IRibbonContainedControl interface 3914
about IRibbonContainedControl interface 3914
ContainedControl 3915
GetContainedControl 3915
IRibbonContainedControl Members 3914
IRibbonContainedControl methods 3915
IRibbonContainedControl properties 3915
SetContainedControl 3915
IRibbonControl interface 3916
about IRibbonControl interface 3916
CanAddToQuickAccessToolbar 3916
IRibbonControl Members 3916
IRibbonControl methods 3916
IRibbonGroup interface 3568

about IRibbonGroup interface 3568
AlignGroupControls 3568
CreateControl 3569
IRibbonGroup Members 3568
IRibbonGroup methods 3568
StyleChanged 3569
IRibbonGroupDropdown interface 3917
about IRibbonGroupDropdown interface 3917
CloseUp 3917
GetParentControl 3918
GetTopPosition 3918
IRibbonGroupDropdown Members 3917
IRibbonGroupDropdown methods 3917
IRibbonKeyTip interface 3918
about IRibbonKeyTip interface 3918
GetKeyTip 3919
HandleKeyTip 3920
HideKeyTip 3920
IRibbonKeyTip Members 3919
IRibbonKeyTip methods 3919
IRibbonKeyTip properties 3922
KeyTip 3922
KeyTipClick 3920
SetKeyTip 3921
ShowKeyTip 3921
IRibbonQuickAccessToolbar interface 3569
about IRibbonQuickAccessToolbar interface 3569
DoUpdateQATWidth 3570
IRibbonQuickAccessToolbar Members 3570
IRibbonQuickAccessToolbar methods 3570
IRibbonSplitButton interface 3922
about IRibbonSplitButton interface 3922
IRibbonSplitButton Members 3923
IRibbonSplitButton methods 3923
UpdateSplitBounds 3923
IRIOAccess interface 4068
about IRIOAccess interface 4068
GetRIO 4069
IRIOAccess Members 4068
IRIOAccess methods 4069
IRIOAccess properties 4069
RIO 4069
IsA
IStub interface 2814
IsActiveLinked
TListActionLink class 1039
ISAPIApp 959
ISAPIApp namespace 959
Classes 960
Functions 965
Variables 967
ISAPIApp.DispatchThread 967
ISAPIApp.DispatchThread variable 967
ISAPIApp.GetExtensionVersion 965
ISAPIApp.GetExtensionVersion function 965
ISAPIApp.HttpExtensionProc 965
ISAPIApp.HttpExtensionProc function 965
ISAPIApp.TerminateExtension 966
ISAPIApp.TerminateExtension function 966
ISAPIApp.TISAPIApplication 960
ISAPIApp.TISAPIApplication.Create 961
ISAPIApp.TISAPIApplication.Destroy 961
ISAPIApp.TISAPIApplication.GetExtensionVersion 962
ISAPIApp.TISAPIApplication.HttpExtensionProc 962
ISAPIApp.TISAPIApplication.TerminateExtension 963
ISAPIApp.TISAPIFactory 963
ISAPIApp.TISAPIFactory.Create 963
ISAPIApp.TISAPIFactory.NewRequest 964
ISAPIApp.TISAPIFactory.NewResponse 964
IsapiHTTP 969
IsapiHTTP namespace 969
Classes 970
IsapiHTTP.TISAPIRequest 970
IsapiHTTP.TISAPIRequest.Create 971
IsapiHTTP.TISAPIRequest.ECB 976
IsapiHTTP.TISAPIRequest.GetDateVariable 972
IsapiHTTP.TISAPIRequest.GetFieldByName 972
IsapiHTTP.TISAPIRequest.GetIntegerVariable 973

IsapiHTTP.TISAPIRequest.GetStringVariable 973
IsapiHTTP.TISAPIRequest.ReadClient 974
IsapiHTTP.TISAPIRequest.ReadString 974
IsapiHTTP.TISAPIRequest.TranslateURI 975
IsapiHTTP.TISAPIRequest.WriteClient 975
IsapiHTTP.TISAPIRequest.WriteHeaders 975
IsapiHTTP.TISAPIRequest.WriteString 976
IsapiHTTP.TISAPIResponse 977
IsapiHTTP.TISAPIResponse.Create 978
IsapiHTTP.TISAPIResponse.GetContent 979
IsapiHTTP.TISAPIResponse.GetDateVariable 979
IsapiHTTP.TISAPIResponse.GetIntegerVariable 980
IsapiHTTP.TISAPIResponse.GetLogMessage 980
IsapiHTTP.TISAPIResponse.GetStatusCode 980
IsapiHTTP.TISAPIResponse.GetStringVariable 981
IsapiHTTP.TISAPIResponse.InitResponse 981
IsapiHTTP.TISAPIResponse.SendRedirect 982
IsapiHTTP.TISAPIResponse.SendResponse 982
IsapiHTTP.TISAPIResponse.SendStream 982
IsapiHTTP.TISAPIResponse.Sent 983
IsapiHTTP.TISAPIResponse.SetContent 983
IsapiHTTP.TISAPIResponse.SetDateVariable 983
IsapiHTTP.TISAPIResponse.SetIntegerVariable 984
IsapiHTTP.TISAPIResponse.SetLogMessage 984
IsapiHTTP.TISAPIResponse.SetStatusCode 984
IsapiHTTP.TISAPIResponse.SetStringVariable 985
IsAutoCheckLinked
TMenuActionLink class 1355
IsButton
TRibbonBaseButtonControl class 3624
IsCallerInRole
IObjectContext interface 2422
TMtsAutoObject class 2391
TMtsDataModule class 2408
IsCaptionLinked
TMenuActionLink class 1355
IsCheckedLinked
TMenuActionLink class 1356
IsClassScalar
TRemotableTypeRegistry class 882
IsColumn
TSelectOptionsColumn class 2027
TStatusColumn class 2035
TTextAreaColumn class 2043
TTextColumn class 2051
TWebDataDisplay class 2081
IsCursorOpen
TIBCustomDataSet class 83
IsCustomProperty
TOleControl class 2669
ISecurityProperty interface 2425
about ISecurityProperty interface 2425
GetDirectCallerSID 2426
GetDirectCreatorSID 2426
GetOriginalCallerSID 2426
GetOriginalCreatorSID 2427
ISecurityProperty Members 2426
ISecurityProperty methods 2426
ReleaseSID 2427
IsEnabledLinked
TMenuActionLink class 1356
IsFieldInUse
IDataSetFields interface 2207
TXMLDisplayGroup class 2184
IsGroupIndexLinked
TMenuActionLink class 1356
ISharedProperty interface 2427
about ISharedProperty interface 2427
Get_Value 2428
ISharedProperty Members 2427
ISharedProperty methods 2428
ISharedProperty properties 2428
Set_Value 2428
Value 2429
ISharedPropertyGroup interface 2429
about ISharedPropertyGroup interface 2429
CreateProperty 2430
CreatePropertyByPosition 2430

Get_PropertyByName 2430
Get_PropertyByPosition 2430
ISharedPropertyGroup Members 2429
ISharedPropertyGroup methods 2429
ISharedPropertyGroup properties 2431
PropertyByName 2431
PropertyByPosition 2431
ISharedPropertyGroupManager interface 2431
_NewEnum 2433
about ISharedPropertyGroupManager interface 2431
CreatePropertyGroup 2432
Get__NewEnum 2432
Get_Group 2433
Group 2433
ISharedPropertyGroupManager Members 2432
ISharedPropertyGroupManager methods 2432
ISharedPropertyGroupManager properties 2433
IsHelpContextLinked
TMenuActionLink class 1357
IsHintLinked
TMenuActionLink class 1357
IsImageIndexLinked
TMenuActionLink class 1358
IsImagesLinked
TListActionLink class 1039
IsInprocServer
TAutomation class 2506
IsInSplit
TRibbonSplitMenuItem class 3900
IsInTransaction
IObjectContext interface 2422
TMtsAutoObject class 2392
TMtsDataModule class 2408
ISkeleton interface 2809
about ISkeleton interface 2809
GetImplementation 2810
GetReplyBuffer 2810
ISkeleton Members 2809
ISkeleton methods 2810
ISkeletonObject interface 2810
about ISkeletonObject interface 2810
Execute 2811
GetImplementation 2811
GetSkeleton 2811
ISkeletonObject Members 2811
ISkeletonObject methods 2811
isLarge enumeration member 3572
IsLine
TMenuItem class 1385
IsLocal
ICorbaObj interface 2796
IsMasked
TCustomMaskEdit class 1096
isMedium enumeration member 3572
IsMultipleRecordView
IWebDisplay interface 2227
IXMLDisplay interface 2230
IsNested
TUpdateTree class 3211
IsNodeAVarArray
TSOAPDomConv class 2773
IsNull
TIBXSQLVAR class 553
IsNullable
TIBXSQLVAR class 553
ISOAPHeaders interface 923
about ISOAPHeaders interface 923
Get 925
Get method 924
GetOwnsSentHeaders 925
ISOAPHeaders Members 924
ISOAPHeaders methods 924
ISOAPHeaders properties 927
OwnsSentHeaders 927
Send 926
SendAt 926
SendCount 927
SetOwnsSentHeaders 927

IsObjectWriting
TSOAPDomConv class 2773
IsOnExecuteLinked
TMenuActionLink class 1358
IsOwnerDraw
TMenu class 1339
IsPanning
TPanningWindow class 703
IsParamInUse
IQueryFields interface 2221
TQueryFieldGroup class 1970
IsPropPageProperty
TOleControl class 2669
IsQueryField
TQueryRadioGroup class 1980
TQuerySelectOptions class 1992
TQueryText class 2004
TQueryTextArea class 2015
TWebDataDisplay class 2081
IsReadOnly
TIBDataBase class 192
IsRibbonAvailable
TRibbonBaseButtonControl class 3624
IsRightToLeft
TMenu class 1339
IsSecurityEnabled
IObjectContext interface 2422
TMtsAutoObject class 2392
TMtsDataModule class 2409
IsSequenced
TIBCustomDataSet class 83
IsServiceRunning
TIBControlService class 424
IsShortCut
TMenu class 1339
IsShortCutLinked
TMenuActionLink class 1358
isSmall enumeration member 3572
IStatusField interface 2223
about IStatusField interface 2223
IStub interface 2812
about IStub interface 2812
CreateRequest 2813
Dispatch 2813
GetInterface 2813
Hash 2813
Invoke 2814
IsA 2814
IStub Members 2812
IStub methods 2812
NonExistent 2814
RepositoryID 2814
SetPrincipal 2815
IStubObject interface 2815
about IStubObject interface 2815
GetStub 2816
IStubObject Members 2815
IStubObject methods 2815
IsValidChar
TRibbonSpinEdit class 3841
IsVisible
TOutlineNode class 2954
IsVisibleLinked
TMenuActionLink class 1359
Item
TPNGList class 2999
TPNGPointerList class 3002
ItemCount
TCustomOutline class 2864
TCustomTabData class 3457
ItemFromClass
TPNGList class 2998
ItemHeight
TCustomOutline class 2864
TOutline class 2888
ItemIndex
TCustomListAction class 1025
TStaticListAction class 1059

TVirtualListAction class 1070
Items
TCustomOutline class 2865
TCustomRibbonComboBox class 3592
TCustomRibbonGroup class 3423
TCustomRibbonQuickAccessToolbar class 3452
TCustomStaticListAction class 1032
TFieldRadioGroup class 1881
TFieldSelectOptions class 1891
TIBExtract class 281
TKeyTipList class 3595
TListControlItems class 1052
TMenu class 1346
TMenuItem class 1411
TOptionItems class 3940
TOutlineNode class 2954
TQueryRadioGroup class 1984
TQuerySelectOptions class 1996
TRibbonComboBox class 3635
TRibbonGalleryBar class 3991
TRibbonTabs class 3564
TSelectOptionsColumn class 2030
TServerCollection class 2470
TStaticListAction class 1059
TWebListInput class 2120
ItemSeparator
TCustomOutline class 2865
TOutline class 2889
ItemsField
TFieldRadioGroup class 1881
TFieldSelectOptions class 1891
TQueryRadioGroup class 1985
TQuerySelectOptions class 1997
TSelectOptionsColumn class 2031
TWebListInput class 2120
ItemsPerRow
TCustomRibbonGalleryBar class 3988
TRibbonGalleryBar class 3992
itImage enumeration member 687
itMask enumeration member 687
ITransactionContext interface 2433
Abort 2434
about ITransactionContext interface 2433
Commit 2434
CreateInstance 2435
ITransactionContext Members 2434
ITransactionContext methods 2434
ITransactionContextEx interface 2435
Abort 2436
about ITransactionContextEx interface 2435
Commit 2436
CreateInstance 2436
ITransactionContextEx Members 2435
ITransactionContextEx methods 2435
ITypeCode interface 2816
about ITypeCode interface 2816
ContentType 2817
DefaultIndex 2817
DiscriminatorType 2817
Equal 2818
Id 2818
ITypeCode Members 2816
ITypeCode methods 2817
Kind 2818
Length 2818
MemberCount 2819
MemberLabel 2819
MemberName 2819
MemberType 2819
Name 2819
Parameter 2820
ParameterCount 2820
IValidateField interface 2223
about IValidateField interface 2223
IValidateField Members 2223
IValidateField methods 2223
ValidateField 2224
IValidateFields interface 2224

about IValidateFields interface 2224
EnableValidateFields 2226
GetEnableValidateFields 2225
IValidateFields Members 2224
IValidateFields methods 2225
IValidateFields properties 2225
SetEnableValidateFields 2225
ValidateFields 2225
IVCLFrameForm interface 2615
about IVCLFrameForm interface 2615
AddContainer 2616
ClearBorderSpace 2616
Form 2616
IVCLFrameForm Members 2615
IVCLFrameForm methods 2615
RemoveContainer 2616
IWebDisplay interface 2226
about IWebDisplay interface 2226
GetIsMultipleRecordView 2227
IsMultipleRecordView 2227
IWebDisplay Members 2226
IWebDisplay methods 2226
IWebDisplay properties 2227
IWebDisplayReference interface 2227
about IWebDisplayReference interface 2227
GetWebDisplayComponent 2228
IWebDisplayReference Members 2228
IWebDisplayReference methods 2228
IWebDisplayReference properties 2228
WebDisplayComponent 2228
IXMLDisplay interface 2229
about IXMLDisplay interface 2229
GetIsMultipleRecordView 2229
GetXMLDisplayName 2230
GetXMLRowSetName 2230
IsMultipleRecordView 2230
IXMLDisplay Members 2229
IXMLDisplay methods 2229
IXMLDisplay properties 2230
XMLDisplayName 2231
XMLRowSetName 2231

J

Java 987
Java namespace 987
Java.Sql 988
Java.Sql namespace 988
Classes 988
Java.Sql.SQLException 988
Java.Sql.SQLException.ErrorCode 989
Java.Sql.SQLException.SQLException 989
Java.Sql.SQLException.SQLState 990
jf24Bit enumeration member 1012
jf8Bit enumeration member 1012
jpBestQuality enumeration member 1012
jpBestSpeed enumeration member 1012
jpeg 991
jpeg namespace 991
Classes 992
Structs, Records, Enums 1011
Types 1014
Variables 1015
jpeg.JPEGDefaults 1015
jpeg.JPEGDefaults variable 1015
jpeg.TJPEGData 992
jpeg.TJPEGData.Destroy 993
jpeg.TJPEGData.FreeHandle 993
jpeg.TJPEGDefaults 1011
jpeg.TJPEGDefaults record 1011
jpeg.TJPEGImage 994
jpeg.TJPEGImage.Assign 998
jpeg.TJPEGImage.AssignTo 999
jpeg.TJPEGImage.Bitmap 1008
jpeg.TJPEGImage.Changed 1000
jpeg.TJPEGImage.Compress 1000
jpeg.TJPEGImage.CompressionQuality 1008
jpeg.TJPEGImage.Create 995
jpeg.TJPEGImage.Destroy 997

jpeg.TJPEGImage.DIBNeeded 1000
jpeg.TJPEGImage.Draw 1001
jpeg.TJPEGImage.Equals 1001
jpeg.TJPEGImage.FreeBitmap 1001
jpeg.TJPEGImage.GetEmpty 1001
jpeg.TJPEGImage.GetHeight 1002
jpeg.TJPEGImage.GetPalette 1002
jpeg.TJPEGImage.GetWidth 1002
jpeg.TJPEGImage.Grayscale 1008
jpeg.TJPEGImage.JPEGNeeded 1003
jpeg.TJPEGImage.LoadFromClipboardFormat 1003
jpeg.TJPEGImage.LoadFromStream 1004
jpeg.TJPEGImage.NewBitmap 1004
jpeg.TJPEGImage.NewImage 1004
jpeg.TJPEGImage.Performance 1008
jpeg.TJPEGImage.PixelFormat 1009
jpeg.TJPEGImage.ProgressiveDisplay 1009
jpeg.TJPEGImage.ProgressiveEncoding 1010
jpeg.TJPEGImage.ReadData 1004
jpeg.TJPEGImage.ReadStream 1005
jpeg.TJPEGImage.SaveToClipboardFormat 1005
jpeg.TJPEGImage.SaveToStream 1005
jpeg.TJPEGImage.Scale 1010
jpeg.TJPEGImage.SetHeight 1006
jpeg.TJPEGImage.SetPalette 1006
jpeg.TJPEGImage.SetWidth 1006
jpeg.TJPEGImage.Smoothing 1010
jpeg.TJPEGImage.WriteData 1007
jpeg.TJPEGPerformance 1012
jpeg.TJPEGPerformance enumeration 1012
jpeg.TJPEGPixelFormat 1012
jpeg.TJPEGPixelFormat enumeration 1012
jpeg.TJPEGQualityRange 1014
jpeg.TJPEGQualityRange type 1014
jpeg.TJPEGScale 1013
jpeg.TJPEGScale enumeration 1013
JPEGNeeded
TJPEGImage class 1003
jsBind enumeration member 2297
jsDB enumeration member 2297
jsDisp enumeration member 2297
jsDom enumeration member 2297
jsEighth enumeration member 1013
jsErrDisp enumeration member 2297
jsFullSize enumeration member 1013
jsHalf enumeration member 1013
jsQuarter enumeration member 1013
jsShow enumeration member 2297

K

Key
TIBLicensingService class 437
TLicenseInfo class 468
KeyDown
TCustomMaskEdit class 1087
TCustomOutline class 2855
TCustomRibbon class 3383
TCustomRibbonComboBox class 3587
TMediaPlayer class 2317
TOleContainer class 2543
TRibbonSpinButton class 3798
TRibbonSpinEdit class 3842
KeyExists
TRegistry class 3266
KeyPress
TCustomMaskEdit class 1088
TCustomOutline class 2855
TCustomRibbonComboBox class 3587
TRibbonSpinEdit class 3842
KeyTip
IRibbonKeyTip interface 3922
TApplicationMenu class 3355
TCustomRibbonGroup class 3423
TCustomRibbonPage class 3438
TOptionItem class 3937
TRibbonApplicationMenuButton class 3612
TRibbonGroup class 3505
TRibbonGroupCollapsedControl class 3518

TRibbonPage class 3543
TRibbonTabItem class 3558
KeyTipClick
IRibbonKeyTip interface 3920
TCustomRibbonGroup class 3416
TRibbonApplicationMenuButton class 3609
TRibbonBaseButtonControl class 3624
TRibbonComboControl class 3692
TRibbonCustomControl class 3704
TRibbonDropDownButton class 3719
TRibbonGroupCollapsedControl class 3512
TRibbonLargeSplitButton class 3750
TRibbonSmallSplitButton class 3794
KeyTipForm
TRibbonTabItem class 3558
KeyTipsVisible
TCustomRibbon class 3397
KeyUp
TCustomMaskEdit class 1088
KeyViolCount
TIBBatchMove class 32
KeyViolTableName
TIBBatchMove class 33
KillShadows enumeration member 485
Kind
ITypeCode interface 2818
ktFull enumeration member 3926
ktNone enumeration member 3926
ktpApplicationButton enumeration member 3927
ktPartial enumeration member 3926
ktpGroupA enumeration member 3927
ktpGroupB enumeration member 3927
ktpGroupC enumeration member 3927
ktpGroupD enumeration member 3927
ktpQuickAccessToolbar enumeration member 3927
ktpTabs enumeration member 3927
ktsCollapsedGroup enumeration member 3928
ktsControls enumeration member 3928
ktsTabs enumeration member 3928

L

LabelContent
TWebDataDisplay class 2082
LastName
TIBSecurityService class 449
TUserInfo class 474
LastVisibleTabIndex
TCustomTabData class 3457
LayoutAttributes
TWebDataDisplay class 2085
LazyWrite
TRegistry class 3296
Left
TRibbonApplicationMenuBar class 3955
LeftTabScrollButtonRect
TCustomTabData class 3457
Length
ITypeCode interface 2818
TMediaPlayer class 2350
Level
TOutlineNode class 2954
License enumeration member 479
LicenseAdd enumeration member 478
LicensedUsers
TLicenseInfo class 468
LicenseInfo
TIBServerProperties class 457
LicenseMask
TLicenseMaskInfo class 469
LicenseMask enumeration member 479
LicenseMaskInfo
TIBServerProperties class 457
LicenseRemove enumeration member 478
LimboState enumeration member 484
LimboTransactionInfo
TIBValidationService class 465
LimboTransactionInfoCount
TIBValidationService class 465

LimboTransactions enumeration member 485
Lines
TCustomOutline class 2866
TOutline class 2889
TQueryTextArea class 2019
Linked
TOleContainer class 2567
List
TOutlineNode class 2955
ListActns 1017
ListActns namespace 1017
Classes 1019
Structs, Records, Enums 1074
Types 1075
ListActns.TCustomListAction 1019
ListActns.TCustomListAction.Active 1024
ListActns.TCustomListAction.Count 1024
ListActns.TCustomListAction.Create 1020
ListActns.TCustomListAction.ExecuteTarget 1021
ListActns.TCustomListAction.GetCount 1021
ListActns.TCustomListAction.GetString 1022
ListActns.TCustomListAction.HandlesTarget 1022
ListActns.TCustomListAction.Images 1025
ListActns.TCustomListAction.ItemIndex 1025
ListActns.TCustomListAction.Loaded 1023
ListActns.TCustomListAction.Loading 1026
ListActns.TCustomListAction.OnGetItemCount 1027
ListActns.TCustomListAction.OnItemSelected 1027
ListActns.TCustomListAction.SetString 1023
ListActns.TCustomListAction.Strings 1026
ListActns.TCustomStaticListAction 1027
ListActns.TCustomStaticListAction.Clients 1032
ListActns.TCustomStaticListAction.Count 1032
ListActns.TCustomStaticListAction.Create 1029
ListActns.TCustomStaticListAction.Destroy 1029
ListActns.TCustomStaticListAction.GetCount 1030
ListActns.TCustomStaticListAction.GetItem 1030
ListActns.TCustomStaticListAction.GetItemClass 1030
ListActns.TCustomStaticListAction.GetString 1031
ListActns.TCustomStaticListAction.Items 1032
ListActns.TCustomStaticListAction.OnGetItem 1033
ListActns.TCustomStaticListAction.SetString 1031
ListActns.TCustomVirtualListAction 1034
ListActns.TCustomVirtualListAction.Count 1036
ListActns.TCustomVirtualListAction.GetItem 1035
ListActns.TCustomVirtualListAction.GetString 1035
ListActns.TCustomVirtualListAction.OnGetItem 1037
ListActns.TGetItemCountEvent 1075
ListActns.TGetItemCountEvent type 1075
ListActns.TGetItemEvent 1075
ListActns.TGetItemEvent type 1075
ListActns.TGetVirtualItemEvent 1076
ListActns.TGetVirtualItemEvent type 1076
ListActns.TItemSelectedEvent 1076
ListActns.TItemSelectedEvent type 1076
ListActns.TListActionLink 1037
ListActns.TListActionLink.AddItem 1039
ListActns.TListActionLink.IsActiveLinked 1039
ListActns.TListActionLink.IsImagesLinked 1039
ListActns.TListActionLink.RefreshControl 1040
ListActns.TListActionLink.SetAction 1040
ListActns.TListActionLink.SetActive 1040
ListActns.TListActionLink.SetImages 1040
ListActns.TListActionLink.SetItemIndex 1041
ListActns.TListCompareEvent 1076
ListActns.TListCompareEvent type 1076
ListActns.TListControlItem 1041
ListActns.TListControlItem.Assign 1044
ListActns.TListControlItem.Caption 1046
ListActns.TListControlItem.Changed 1044
ListActns.TListControlItem.Create 1042
ListActns.TListControlItem.Data 1047
ListActns.TListControlItem.FCaption 1043
ListActns.TListControlItem.FData 1043
ListActns.TListControlItem.FImageIndex 1043
ListActns.TListControlItem.GetDisplayName 1045
ListActns.TListControlItem.ImageIndex 1047
ListActns.TListControlItem.SetCaption 1045

ListActns.TListControlItem.SetData 1045
ListActns.TListControlItem.SetImageIndex 1046
ListActns.TListControlItemClass 1077
ListActns.TListControlItemClass type 1077
ListActns.TListControlItems 1047
ListActns.TListControlItems.Add 1049
ListActns.TListControlItems.CaseSensitive 1052
ListActns.TListControlItems.CompareItems 1050
ListActns.TListControlItems.Create 1049
ListActns.TListControlItems.CustomSort 1050
ListActns.TListControlItems.Items 1052
ListActns.TListControlItems.OnCompare 1054
ListActns.TListControlItems.Sort 1051
ListActns.TListControlItems.SortType 1053
ListActns.TListControlItems.Update 1051
ListActns.TListItemsCompare 1077
ListActns.TListItemsCompare type 1077
ListActns.TListItemsSortType 1074
ListActns.TListItemsSortType enumeration 1074
ListActns.TStaticListAction 1054
ListActns.TStaticListAction.Active 1056
ListActns.TStaticListAction.Caption 1057
ListActns.TStaticListAction.Enabled 1057
ListActns.TStaticListAction.HelpContext 1058
ListActns.TStaticListAction.Hint 1058
ListActns.TStaticListAction.Images 1058
ListActns.TStaticListAction.ItemIndex 1059
ListActns.TStaticListAction.Items 1059
ListActns.TStaticListAction.OnGetItem 1061
ListActns.TStaticListAction.OnHint 1061
ListActns.TStaticListAction.OnItemSelected 1062
ListActns.TStaticListAction.OnUpdate 1062
ListActns.TStaticListAction.SecondaryShortCuts 1060
ListActns.TStaticListAction.ShortCut 1060
ListActns.TStaticListAction.Visible 1060
ListActns.TStaticListItems 1063
ListActns.TStaticListItems.Notify 1065
ListActns.TStaticListItems.Update 1065
ListActns.TVirtualListAction 1066
ListActns.TVirtualListAction.Active 1068
ListActns.TVirtualListAction.Caption 1068
ListActns.TVirtualListAction.Enabled 1068
ListActns.TVirtualListAction.HelpContext 1069
ListActns.TVirtualListAction.Hint 1069
ListActns.TVirtualListAction.Images 1070
ListActns.TVirtualListAction.ItemIndex 1070
ListActns.TVirtualListAction.OnGetItem 1072
ListActns.TVirtualListAction.OnGetItemCount 1072
ListActns.TVirtualListAction.OnHint 1073
ListActns.TVirtualListAction.OnItemSelected 1073
ListActns.TVirtualListAction.SecondaryShortCuts 1070
ListActns.TVirtualListAction.ShortCut 1071
ListActns.TVirtualListAction.Visible 1071
ListAllTables
TIBExtract class 276
ListCheck
TIBExtract class 277
ListCreateDb
TIBExtract class 277
ListData
TIBExtract class 277
ListDomains
TIBExtract class 277
ListException
TIBExtract class 278
ListFilters
TIBExtract class 278
ListForeign
TIBExtract class 278
ListFunctions
TIBExtract class 278
ListGenerators
TIBExtract class 278
ListGrants
TIBExtract class 279
ListIndex
TIBExtract class 279
ListProcs

TIBExtract class 279
ListRoles
TIBExtract class 279
ListTriggers
TIBExtract class 280
ListViews
TIBExtract class 280
LiveMode
TIBCustomDataSet class 101
TIBDataSet class 128
TIBQuery class 389
TIBTable class 619
lmDelete enumeration member 152
lmInsert enumeration member 152
lmModify enumeration member 152
lmRefresh enumeration member 152
LoadBalanced
TSimpleObjectBroker class 2480
Loaded
TCustomListAction class 1023
TCustomOutline class 2856
TCustomRibbon class 3384
TCustomRibbonPage class 3432
TIBCustomDataSet class 84
TIBCustomService class 428
TIBDataBase class 186
TIBSecurityService class 447
TIBTransaction class 212
TMediaPlayer class 2317
TMenu class 1340
TMenuItem class 1385
TOleServer class 2716
TRibbonSpinButton class 3799
LoadFromClipboardFormat
TJPEGImage class 1003
LoadFromFile
TCustomOutline class 2856
TIBBlobStream class 46
TIBXSQLVAR class 547
TOleContainer class 2544
LoadFromStream
TCustomOutline class 2857
TIBBlobStream class 46
TIBXSQLVAR class 547
TJPEGImage class 1004
TOleContainer class 2545
TSimpleObjectBroker class 2479
Loading
TCustomListAction class 1026
LoadKey
TRegistry class 3268
LoadObject
TSOAPDomConv class 2773
LoadSkin
TCustomRibbonSkin class 4022
Local enumeration member 479
Locate
TIBCustomDataSet class 84
TIBTable class 611
LocateNext
TIBCustomDataSet class 85
LocateRecord
TBaseProvider class 3086
TDataSetProvider class 3157
Lock
TRemotableTypeRegistry class 883
LockFileLocation
TConfigParams class 403
LockInPlaceActive
TOleControl class 2670
Login
TIBCustomService class 429
LoginPrompt
TDispatchConnection class 1296
TIBCustomService class 430
TIBDataBase class 192
LogUpdateError
TCustomResolver class 3133

LogUpdateRecord
TCustomResolver class 3134
Lookup
TIBCustomDataSet class 85
lrDefaultColor enumeration member 687
lrDefaultSize enumeration member 687
lrFromFile enumeration member 687
lrMap3DColors enumeration member 687
lrMonoChrome enumeration member 687
lrTransparent enumeration member 687
lsButtons enumeration member 2238
lsFields enumeration member 2238
lsUnknown enumeration member 2238
ltFile enumeration member 4020
ltResource enumeration member 4020

M

maAutomatic enumeration member 1462
maBottomToTop enumeration member 1461
MainComponents
TInstallOptions class 326
MainWndProc
TPopupList class 1429
MakeAny
IORB interface 2808
MakeArrayNode
TSOAPDomConv class 2774
MakeFault
IOPConvert interface 2733
TOPToSoapDomConvert class 2752
MakeResponse
IOPConvert interface 2734
TOPToSoapDomConvert class 2753
maLeftToRight enumeration member 1461
maManual enumeration member 1462
maNone enumeration member 1461
maParent enumeration member 1462
Mappings
TIBBatchMove class 33
maRightToLeft enumeration member 1461
Marks
TIBDatabaseInfo class 251
Mask 1079
Mask namespace 1079
Classes 1080
Types 1175
Mask.EDBEditError 1080
Mask.TCustomMaskEdit 1080
Mask.TCustomMaskEdit.CheckCursor 1083
Mask.TCustomMaskEdit.Clear 1083
Mask.TCustomMaskEdit.Create 1082
Mask.TCustomMaskEdit.EditCanModify 1084
Mask.TCustomMaskEdit.EditMask 1092
Mask.TCustomMaskEdit.EditText 1095
Mask.TCustomMaskEdit.GetFirstEditChar 1084
Mask.TCustomMaskEdit.GetLastEditChar 1085
Mask.TCustomMaskEdit.GetMaxChars 1085
Mask.TCustomMaskEdit.GetNextEditChar 1086
Mask.TCustomMaskEdit.GetPriorEditChar 1086
Mask.TCustomMaskEdit.GetSel 1087
Mask.TCustomMaskEdit.GetTextLen 1087
Mask.TCustomMaskEdit.IsMasked 1096
Mask.TCustomMaskEdit.KeyDown 1087
Mask.TCustomMaskEdit.KeyPress 1088
Mask.TCustomMaskEdit.KeyUp 1088
Mask.TCustomMaskEdit.MaskState 1096
Mask.TCustomMaskEdit.MaxLength 1096
Mask.TCustomMaskEdit.ReformatText 1089
Mask.TCustomMaskEdit.Reset 1089
Mask.TCustomMaskEdit.SetCursor 1090
Mask.TCustomMaskEdit.SetSel 1090
Mask.TCustomMaskEdit.Text 1097
Mask.TCustomMaskEdit.Validate 1091
Mask.TCustomMaskEdit.ValidateEdit 1091
Mask.TCustomMaskEdit.ValidateError 1091
Mask.TMaskEdit 1100
Mask.TMaskEdit.Align 1104
Mask.TMaskEdit.Alignment 1107

Mask.TMaskEdit.Anchors 1108
Mask.TMaskEdit.AutoSelect 1108
Mask.TMaskEdit.AutoSize 1109
Mask.TMaskEdit.BevelEdges 1110
Mask.TMaskEdit.BevelInner 1110
Mask.TMaskEdit.BevelKind 1111
Mask.TMaskEdit.BevelOuter 1111
Mask.TMaskEdit.BevelWidth 1112
Mask.TMaskEdit.BiDiMode 1112
Mask.TMaskEdit.BorderStyle 1112
Mask.TMaskEdit.CharCase 1113
Mask.TMaskEdit.Color 1113
Mask.TMaskEdit.Constraints 1114
Mask.TMaskEdit.Ctl3D 1115
Mask.TMaskEdit.DoubleBuffered 1115
Mask.TMaskEdit.DragCursor 1117
Mask.TMaskEdit.DragKind 1117
Mask.TMaskEdit.DragMode 1118
Mask.TMaskEdit.EditMask 1119
Mask.TMaskEdit.Enabled 1122
Mask.TMaskEdit.Font 1123
Mask.TMaskEdit.ImeMode 1124
Mask.TMaskEdit.ImeName 1124
Mask.TMaskEdit.MaxLength 1125
Mask.TMaskEdit.OnChange 1142
Mask.TMaskEdit.OnClick 1142
Mask.TMaskEdit.OnDblClick 1144
Mask.TMaskEdit.OnDragDrop 1144
Mask.TMaskEdit.OnDragOver 1147
Mask.TMaskEdit.OnEndDock 1150
Mask.TMaskEdit.OnEndDrag 1151
Mask.TMaskEdit.OnEnter 1152
Mask.TMaskEdit.OnExit 1154
Mask.TMaskEdit.OnKeyDown 1158
Mask.TMaskEdit.OnKeyPress 1162
Mask.TMaskEdit.OnKeyUp 1163
Mask.TMaskEdit.OnMouseActivate 1165
Mask.TMaskEdit.OnMouseDown 1166
Mask.TMaskEdit.OnMouseEnter 1168
Mask.TMaskEdit.OnMouseLeave 1168
Mask.TMaskEdit.OnMouseMove 1168
Mask.TMaskEdit.OnMouseUp 1171
Mask.TMaskEdit.OnStartDock 1173
Mask.TMaskEdit.OnStartDrag 1173
Mask.TMaskEdit.ParentBiDiMode 1125
Mask.TMaskEdit.ParentColor 1126
Mask.TMaskEdit.ParentCtl3D 1126
Mask.TMaskEdit.ParentDoubleBuffered 1127
Mask.TMaskEdit.ParentFont 1127
Mask.TMaskEdit.ParentShowHint 1127
Mask.TMaskEdit.PasswordChar 1128
Mask.TMaskEdit.PopupMenu 1128
Mask.TMaskEdit.ReadOnly 1133
Mask.TMaskEdit.ShowHint 1134
Mask.TMaskEdit.TabOrder 1135
Mask.TMaskEdit.TabStop 1136
Mask.TMaskEdit.Text 1136
Mask.TMaskEdit.TextHint 1139
Mask.TMaskEdit.Visible 1139
Mask.TMaskedState 1175
Mask.TMaskedState type 1175
Masked
    TCustomImageList class 683
Masks 1177
Masks namespace 1177
    Classes 1178
    Functions 1182
Masks.EMaskException 1178
Masks.MatchesMask 1182
Masks.MatchesMask function 1182
Masks.TMask 1178
Masks.TMask.Create 1179
Masks.TMask.Destroy 1180
Masks.TMask.Matches 1180
MaskState
    TCustomMaskEdit class 1096
MaskUtils 1183
MaskUtils namespace 1183

Constants 1198
Functions 1185
Structs, Records, Enums 1192
Types 1193
Variables 1196
MaskUtils.DefaultBlank 1196
MaskUtils.DefaultBlank variable 1196
MaskUtils.FormatMaskText 1185
MaskUtils.FormatMaskText function 1185
MaskUtils.IsLiteralChar 1187
MaskUtils.IsLiteralChar function 1187
MaskUtils.MaskDoFormatText 1187
MaskUtils.MaskDoFormatText function 1187
MaskUtils.MaskFieldSeparator 1196
MaskUtils.MaskFieldSeparator variable 1196
MaskUtils.MaskGetCharType 1187
MaskUtils.MaskGetCharType function 1187
MaskUtils.MaskGetCurrentDirectives 1188
MaskUtils.MaskGetCurrentDirectives function 1188
MaskUtils.MaskGetFldSeparator 1188
MaskUtils.MaskGetFldSeparator function 1188
MaskUtils.MaskGetMaskBlank 1189
MaskUtils.MaskGetMaskBlank function 1189
MaskUtils.MaskGetMaskSave 1189
MaskUtils.MaskGetMaskSave function 1189
MaskUtils.MaskIntlLiteralToChar 1189
MaskUtils.MaskIntlLiteralToChar function 1189
MaskUtils.MaskNoSave 1197
MaskUtils.MaskNoSave variable 1197
MaskUtils.MaskOffsetToOffset 1190
MaskUtils.MaskOffsetToOffset function 1190
MaskUtils.MaskOffsetToWideOffset 1190
MaskUtils.MaskOffsetToWideOffset function 1190
MaskUtils.mDirLiteral 1198
MaskUtils.mDirLiteral constant 1198
MaskUtils.mDirLowerCase 1198
MaskUtils.mDirLowerCase constant 1198
MaskUtils.mDirReverse 1199
MaskUtils.mDirReverse constant 1199
MaskUtils.mDirUpperCase 1199
MaskUtils.mDirUpperCase constant 1199
MaskUtils.mMskAlpha 1200
MaskUtils.mMskAlpha constant 1200
MaskUtils.mMskAlphaNum 1200
MaskUtils.mMskAlphaNum constant 1200
MaskUtils.mMskAlphaNumOpt 1200
MaskUtils.mMskAlphaNumOpt constant 1200
MaskUtils.mMskAlphaOpt 1201
MaskUtils.mMskAlphaOpt constant 1201
MaskUtils.mMskAscii 1201
MaskUtils.mMskAscii constant 1201
MaskUtils.mMskAsciiOpt 1202
MaskUtils.mMskAsciiOpt constant 1202
MaskUtils.mMskDateSeparator 1202
MaskUtils.mMskDateSeparator constant 1202
MaskUtils.mMskNumeric 1202
MaskUtils.mMskNumeric constant 1202
MaskUtils.mMskNumericOpt 1203
MaskUtils.mMskNumericOpt constant 1203
MaskUtils.mMskNumSymOpt 1203
MaskUtils.mMskNumSymOpt constant 1203
MaskUtils.mMskTimeSeparator 1204
MaskUtils.mMskTimeSeparator constant 1204
MaskUtils.OffsetToMaskOffset 1191
MaskUtils.OffsetToMaskOffset function 1191
MaskUtils.PadInputLiterals 1191
MaskUtils.PadInputLiterals function 1191
MaskUtils.TEditMask 1193
MaskUtils.TEditMask type 1193
MaskUtils.TMaskCharType 1192
MaskUtils.TMaskCharType enumeration 1192
MaskUtils.TMaskDirectives 1194
MaskUtils.TMaskDirectives type 1194
MaskUtils.TMaskedText 1195
MaskUtils.TMaskedText type 1195
MasterFields
TIBTable class 620
MasterSource

TIBTable class 620
Matches
TMask class 1180
MatchType
TKeyTipMatchItem class 3600
Math 1205
Math namespace 1205
Classes 1209
Constants 1270
Functions 1211
Structs, Records, Enums 1263
Types 1267
Math.ArcCos 1213
Math.ArcCos function 1213
Math.ArcCosh 1214
Math.ArcCosh function 1214
Math.ArcCot 1214
Math.ArcCot function 1214
Math.ArcCotH 1215
Math.ArcCotH function 1215
Math.ArcCsc 1215
Math.ArcCsc function 1215
Math.ArcCscH 1216
Math.ArcCscH function 1216
Math.ArcSec 1216
Math.ArcSec function 1216
Math.ArcSecH 1216
Math.ArcSecH function 1216
Math.ArcSin 1217
Math.ArcSin function 1217
Math.ArcSinh 1217
Math.ArcSinh function 1217
Math.ArcTan2 1218
Math.ArcTan2 function 1218
Math.ArcTanh 1218
Math.ArcTanh function 1218
Math.Ceil 1219
Math.Ceil function 1219
Math.ClearExceptions 1219
Math.ClearExceptions function 1219
Math.CompareValue 1220
Math.CompareValue function 1220
Math.Cosecant 1220
Math.Cosecant function 1220
Math.Cosh 1221
Math.Cosh function 1221
Math.Cot 1221
Math.Cot function 1221
Math.Cotan 1222
Math.Cotan function 1222
Math.CotH 1222
Math.CotH function 1222
Math.Csc 1223
Math.Csc function 1223
Math.CscH 1223
Math.CscH function 1223
Math.CycleToDeg 1224
Math.CycleToDeg function 1224
Math.CycleToGrad 1224
Math.CycleToGrad function 1224
Math.CycleToRad 1224
Math.CycleToRad function 1224
Math.DegToCycle 1225
Math.DegToCycle function 1225
Math.DegToGrad 1225
Math.DegToGrad function 1225
Math.DegToRad 1226
Math.DegToRad function 1226
Math.DivMod 1226
Math.DivMod function 1226
Math.DoubleDecliningBalance 1226
Math.DoubleDecliningBalance function 1226
Math.EInvalidArgument 1209
Math.EnsureRange 1227
Math.EnsureRange function 1227
Math.Floor 1227
Math.Floor function 1227
Math.Frexp 1228

Math.Frexp function 1228
Math.FutureValue 1228
Math.FutureValue function 1228
Math.GetExceptionMask 1229
Math.GetExceptionMask function 1229
Math.GetPrecisionMode 1229
Math.GetPrecisionMode function 1229
Math.GetRoundMode 1230
Math.GetRoundMode function 1230
Math.GradToCycle 1231
Math.GradToCycle function 1231
Math.GradToDeg 1231
Math.GradToDeg function 1231
Math.GradToRad 1231
Math.GradToRad function 1231
Math.Hypot 1232
Math.Hypot function 1232
Math.IfThen 1232
Math.IfThen function 1232
Math.Infinity 1270
Math.Infinity constant 1270
Math.InRange 1232
Math.InRange function 1232
Math.InterestPayment 1233
Math.InterestPayment function 1233
Math.InterestRate 1234
Math.InterestRate function 1234
Math.InternalRateOfReturn 1234
Math.InternalRateOfReturn function 1234
Math.IntPower 1235
Math.IntPower function 1235
Math.IsInfinite 1235
Math.IsInfinite function 1235
Math.IsNan 1236
Math.IsNan function 1236
Math.IsZero 1236
Math.IsZero function 1236
Math.Ldexp 1237
Math.Ldexp function 1237
Math.LnXP1 1237
Math.LnXP1 function 1237
Math.Log10 1237
Math.Log10 function 1237
Math.Log2 1238
Math.Log2 function 1238
Math.LogN 1238
Math.LogN function 1238
Math.Max 1239
Math.Max function 1239
Math.MaxComp 1271
Math.MaxComp constant 1271
Math.MaxDouble 1271
Math.MaxDouble constant 1271
Math.MaxExtended 1271
Math.MaxExtended constant 1271
Math.MaxIntValue 1239
Math.MaxIntValue function 1239
Math.MaxSingle 1272
Math.MaxSingle constant 1272
Math.MaxValue 1240
Math.MaxValue function 1240
Math.Mean 1240
Math.Mean function 1240
Math.MeanAndStdDev 1241
Math.MeanAndStdDev function 1241
Math.Min 1241
Math.Min function 1241
Math.MinComp 1272
Math.MinComp constant 1272
Math.MinDouble 1273
Math.MinDouble constant 1273
Math.MinExtended 1273
Math.MinExtended constant 1273
Math.MinIntValue 1242
Math.MinIntValue function 1242
Math.MinSingle 1273
Math.MinSingle constant 1273
Math.MinValue 1242

Math.MinValue function 1242
Math.MomentSkewKurtosis 1243
Math.MomentSkewKurtosis function 1243
Math.NaN 1274
Math.NaN constant 1274
Math.NegativeValue 1274
Math.NegativeValue constant 1274
Math.NegInfinity 1275
Math.NegInfinity constant 1275
Math.NetPresentValue 1243
Math.NetPresentValue function 1243
Math.Norm 1244
Math.Norm function 1244
Math.NumberOfPeriods 1244
Math.NumberOfPeriods function 1244
Math.Payment 1245
Math.Payment function 1245
Math.PeriodPayment 1245
Math.PeriodPayment function 1245
Math.Poly 1245
Math.Poly function 1245
Math.PopnStdDev 1246
Math.PopnStdDev function 1246
Math.PopnVariance 1246
Math.PopnVariance function 1246
Math.PositiveValue 1275
Math.PositiveValue constant 1275
Math.Power 1247
Math.Power function 1247
Math.PresentValue 1247
Math.PresentValue function 1247
Math.RadToCycle 1248
Math.RadToCycle function 1248
Math.RadToDeg 1248
Math.RadToDeg function 1248
Math.RadToGrad 1248
Math.RadToGrad function 1248
Math.RandG 1249
Math.RandG function 1249
Math.RandomFrom 1249
Math.RandomFrom function 1249
Math.RandomRange 1250
Math.RandomRange function 1250
Math.RoundTo 1250
Math.RoundTo function 1250
Math.SameValue 1251
Math.SameValue function 1251
Math.Sec 1252
Math.Sec function 1252
Math.Secant 1252
Math.Secant function 1252
Math.SecH 1253
Math.SecH function 1253
Math.SetExceptionMask 1253
Math.SetExceptionMask function 1253
Math.SetPrecisionMode 1254
Math.SetPrecisionMode function 1254
Math.SetRoundMode 1254
Math.SetRoundMode function 1254
Math.Sign 1255
Math.Sign function 1255
Math.SimpleRoundTo 1255
Math.SimpleRoundTo function 1255
Math.SinCos 1256
Math.SinCos function 1256
Math.Sinh 1256
Math.Sinh function 1256
Math.SLNDepreciation 1257
Math.SLNDepreciation function 1257
Math.StdDev 1257
Math.StdDev function 1257
Math.Sum 1258
Math.Sum function 1258
Math.SumInt 1258
Math.SumInt function 1258
Math.SumOfSquares 1259
Math.SumOfSquares function 1259
Math.SumsAndSquares 1259

Math.SumsAndSquares function 1259
Math.SYDDepreciation 1260
Math.SYDDepreciation function 1260
Math.Tan 1260
Math.Tan function 1260
Math.Tanh 1261
Math.Tanh function 1261
Math.TFPUException 1263
Math.TFPUException enumeration 1263
Math.TFPUExceptionMask 1267
Math.TFPUExceptionMask type 1267
Math.TFPUPrecisionMode 1264
Math.TFPUPrecisionMode enumeration 1264
Math.TFPURoundingMode 1265
Math.TFPURoundingMode enumeration 1265
Math.TotalVariance 1261
Math.TotalVariance function 1261
Math.TPaymentTime 1266
Math.TPaymentTime enumeration 1266
Math.TRoundToEXRangeExtended 1268
Math.TRoundToEXRangeExtended type 1268
Math.TRoundToRange 1268
Math.TRoundToRange type 1268
Math.TValueSign 1268
Math.TValueSign type 1268
Math.Variance 1262
Math.Variance function 1262
Math.ZeroValue 1276
Math.ZeroValue constant 1276
maTopToBottom enumeration member 1461
MaxLength
TCustomMaskEdit class 1096
TMaskEdit class 1125
TRibbonComboBox class 3635
TRibbonSpinEdit class 3853
MaxMemory
TIBDatabaseInfo class 251
MaxValue
TRibbonSpinEdit class 3854
MaxWidth
TQueryText class 2007
mbBarBreak enumeration member 1461
mbBreak enumeration member 1461
mbNone enumeration member 1461
mcDirective enumeration member 1192
mcField enumeration member 1192
mcFieldSeparator enumeration member 1192
mcIntlLiteral enumeration member 1192
mcLiteral enumeration member 1192
mcMask enumeration member 1192
mcMaskOpt enumeration member 1192
mcNone enumeration member 1192
MConnect 1277
MConnect namespace 1277
Classes 1278
Functions 1317
MConnect.GetMIDASAppServerList 1317
MConnect.GetMIDASAppServerList function 1317
MConnect.TCOMConnection 1278
MConnect.TCOMConnection.DoConnect 1280
MConnect.TCOMConnection.SetConnected 1280
MConnect.TCustomObjectBroker 1281
MConnect.TCustomObjectBroker.GetComputerForGUID 1282
MConnect.TCustomObjectBroker.GetComputerForProgID 1283
MConnect.TCustomObjectBroker.GetPortForComputer 1283
MConnect.TCustomObjectBroker.SetConnectStatus 1284
MConnect.TDCOMConnection 1284
MConnect.TDCOMConnection.ComputerName 1287
MConnect.TDCOMConnection.Create 1286
MConnect.TDCOMConnection.DoConnect 1287
MConnect.TDCOMConnection.ObjectBroker 1288
MConnect.TDispatchAppServer 1289
MConnect.TDispatchAppServer.Create 1289
MConnect.TDispatchAppServer.SafeCallException 1290
MConnect.TDispatchConnection 1290
MConnect.TDispatchConnection.AfterConnect 1302
MConnect.TDispatchConnection.AfterDisconnect 1302
MConnect.TDispatchConnection.BeforeConnect 1303

MConnect.TDispatchConnection.BeforeDisconnect 1303
MConnect.TDispatchConnection.Connected 1296
MConnect.TDispatchConnection.Create 1291
MConnect.TDispatchConnection.DoDisconnect 1292
MConnect.TDispatchConnection.GetConnected 1293
MConnect.TDispatchConnection.GetServer 1293
MConnect.TDispatchConnection.GetServerCLSID 1293
MConnect.TDispatchConnection.GetServerList 1294
MConnect.TDispatchConnection.LoginPrompt 1296
MConnect.TDispatchConnection.Notification 1294
MConnect.TDispatchConnection.ObjectBroker 1300
MConnect.TDispatchConnection.OnGetUsername 1304
MConnect.TDispatchConnection.OnLogin 1304
MConnect.TDispatchConnection.ServerGUID 1301
MConnect.TDispatchConnection.ServerName 1301
MConnect.TDispatchConnection.SetConnected 1295
MConnect.TOLEnterpriseConnection 1305
MConnect.TOLEnterpriseConnection.BrokerName 1308
MConnect.TOLEnterpriseConnection.ComputerName 1308
MConnect.TOLEnterpriseConnection.DoConnect 1307
MConnect.TSharedConnection 1309
MConnect.TSharedConnection.AfterConnect 1315
MConnect.TSharedConnection.AfterDisconnect 1315
MConnect.TSharedConnection.BeforeConnect 1315
MConnect.TSharedConnection.BeforeDisconnect 1316
MConnect.TSharedConnection.ChildName 1314
MConnect.TSharedConnection.ConnectEvent 1311
MConnect.TSharedConnection.Create 1310
MConnect.TSharedConnection.Destroy 1311
MConnect.TSharedConnection.DoConnect 1311
MConnect.TSharedConnection.DoDisconnect 1312
MConnect.TSharedConnection.GetConnected 1312
MConnect.TSharedConnection.GetServer 1313
MConnect.TSharedConnection.Notification 1313
MConnect.TSharedConnection.ParentConnection 1314
MeasureItem
TMenuItem class 1386
MemberCount
ITypeCode interface 2819
MemberLabel
ITypeCode interface 2819
MemberName
ITypeCode interface 2819
MemberType
ITypeCode interface 2819
MendDB enumeration member 485
Menu
TApplicationMenu class 3355
TCustomRibbon class 3397
MenuAnimation
TPopupMenu class 1439
MenuChanged
TMainMenu class 1325
TMenu class 1340
TMenuItem class 1386
MenuIndex
TMenuItem class 1413
Menus 1319
Menus namespace 1319
Classes 1321
Constants 1472
Functions 1446
Structs, Records, Enums 1460
Types 1465
Variables 1470
Menus.AnsiSameCaption 1446
Menus.AnsiSameCaption function 1446
Menus.cDialogSuffix 1472
Menus.cDialogSuffix constant 1472
Menus.cHotkeyPrefix 1472
Menus.cHotkeyPrefix constant 1472
Menus.cLineCaption 1472
Menus.cLineCaption constant 1472
Menus.DrawMenuItem 1447
Menus.DrawMenuItem function 1447
Menus.EMenuError 1321
Menus.GetHotkey 1447
Menus.GetHotkey function 1447

Menus.IsAltGRPressed 1447
Menus.IsAltGRPressed function 1447
Menus.NewItem 1448
Menus.NewItem function 1448
Menus.NewLine 1449
Menus.NewLine function 1449
Menus.NewMenu 1450
Menus.NewMenu function 1450
Menus.NewPopupMenu 1451
Menus.NewPopupMenu function 1451
Menus.NewSubMenu 1452
Menus.NewSubMenu function 1452
Menus.PMenuItem 1465
Menus.PMenuItem type 1465
Menus.PopupList 1470
Menus.PopupList variable 1470
Menus.SameCaption 1453
Menus.SameCaption function 1453
Menus.ShortCut 1453
Menus.ShortCut function 1453
Menus.ShortCutFromMessage 1454
Menus.ShortCutFromMessage function 1454
Menus.ShortCutItems 1470
Menus.ShortCutItems variable 1470
Menus.ShortCutToKey 1454
Menus.ShortCutToKey function 1454
Menus.ShortCutToText 1456
Menus.ShortCutToText function 1456
Menus.StripHotkey 1457
Menus.StripHotkey function 1457
Menus.TAdvancedMenuDrawItemEvent 1465
Menus.TAdvancedMenuDrawItemEvent type 1465
Menus.TextToShortCut 1457
Menus.TextToShortCut function 1457
Menus.TFindItemKind 1460
Menus.TFindItemKind enumeration 1460
Menus.TMainMenu 1322
Menus.TMainMenu.AutoHotkeys 1327
Menus.TMainMenu.AutoLineReduction 1328
Menus.TMainMenu.AutoMerge 1328
Menus.TMainMenu.BiDiMode 1329
Menus.TMainMenu.GetHandle 1324
Menus.TMainMenu.GetOle2AcceleratorTable 1324
Menus.TMainMenu.Images 1329
Menus.TMainMenu.MenuChanged 1325
Menus.TMainMenu.Merge 1325
Menus.TMainMenu.OnChange 1331
Menus.TMainMenu.OwnerDraw 1330
Menus.TMainMenu.ParentBiDiMode 1330
Menus.TMainMenu.PopulateOle2Menu 1326
Menus.TMainMenu.SetOle2MenuHandle 1326
Menus.TMainMenu.Unmerge 1326
Menus.TMenu 1331
Menus.TMenu.AdjustBiDiBehavior 1335
Menus.TMenu.AutoHotkeys 1343
Menus.TMenu.AutoLineReduction 1344
Menus.TMenu.BiDiMode 1344
Menus.TMenu.Create 1333
Menus.TMenu.CreateMenuItem 1335
Menus.TMenu.Destroy 1334
Menus.TMenu.DispatchCommand 1335
Menus.TMenu.DispatchPopup 1336
Menus.TMenu.DoBiDiModeChanged 1336
Menus.TMenu.DoChange 1336
Menus.TMenu.DoGetMenuString 1337
Menus.TMenu.FindItem 1337
Menus.TMenu.GetChildren 1338
Menus.TMenu.GetHandle 1338
Menus.TMenu.GetHelpContext 1338
Menus.TMenu.Handle 1345
Menus.TMenu.Images 1345
Menus.TMenu.IsOwnerDraw 1339
Menus.TMenu.IsRightToLeft 1339
Menus.TMenu.IsShortCut 1339
Menus.TMenu.Items 1346
Menus.TMenu.Loaded 1340
Menus.TMenu.MenuChanged 1340
Menus.TMenu.Notification 1341

Menus.TMenu.OnChange 1352
Menus.TMenu.OwnerDraw 1350
Menus.TMenu.ParentBiDiMode 1351
Menus.TMenu.ParentBiDiModeChanged 1341
Menus.TMenu.ProcessMenuChar 1342
Menus.TMenu.SetChildOrder 1342
Menus.TMenu.UpdateItems 1343
Menus.TMenu.WindowHandle 1351
Menus.TMenuActionLink 1352
Menus.TMenuActionLink.AssignClient 1354
Menus.TMenuActionLink.FClient 1354
Menus.TMenuActionLink.IsAutoCheckLinked 1355
Menus.TMenuActionLink.IsCaptionLinked 1355
Menus.TMenuActionLink.IsCheckedLinked 1356
Menus.TMenuActionLink.IsEnabledLinked 1356
Menus.TMenuActionLink.IsGroupIndexLinked 1356
Menus.TMenuActionLink.IsHelpContextLinked 1357
Menus.TMenuActionLink.IsHintLinked 1357
Menus.TMenuActionLink.IsImageIndexLinked 1358
Menus.TMenuActionLink.IsOnExecuteLinked 1358
Menus.TMenuActionLink.IsShortCutLinked 1358
Menus.TMenuActionLink.IsVisibleLinked 1359
Menus.TMenuActionLink.SetAutoCheck 1359
Menus.TMenuActionLink.SetCaption 1359
Menus.TMenuActionLink.SetChecked 1360
Menus.TMenuActionLink.SetEnabled 1360
Menus.TMenuActionLink.SetHelpContext 1361
Menus.TMenuActionLink.SetHint 1361
Menus.TMenuActionLink.SetImageIndex 1361
Menus.TMenuActionLink.SetOnExecute 1362
Menus.TMenuActionLink.SetShortCut 1362
Menus.TMenuActionLink.SetVisible 1362
Menus.TMenuActionLinkClass 1466
Menus.TMenuActionLinkClass type 1466
Menus.TMenuAnimation 1466
Menus.TMenuAnimation type 1466
Menus.TMenuAnimations 1461
Menus.TMenuAnimations enumeration 1461
Menus.TMenuAutoFlag 1467
Menus.TMenuAutoFlag type 1467
Menus.TMenuBreak 1461
Menus.TMenuBreak enumeration 1461
Menus.TMenuChangeEvent 1467
Menus.TMenuChangeEvent type 1467
Menus.TMenuDrawItemEvent 1468
Menus.TMenuDrawItemEvent type 1468
Menus.TMenuHandle 1468
Menus.TMenuHandle type 1468
Menus.TMenuItem 1363
Menus.TMenuItem.Action 1393
Menus.TMenuItem.ActionChange 1369
Menus.TMenuItem.ActionLink 1394
Menus.TMenuItem.Add 1369
Menus.TMenuItem.AdvancedDrawItem 1374
Menus.TMenuItem.AssignTo 1374
Menus.TMenuItem.AutoCheck 1394
Menus.TMenuItem.AutoHotkeys 1394
Menus.TMenuItem.AutoLineReduction 1395
Menus.TMenuItem.Bitmap 1395
Menus.TMenuItem.Break 1396
Menus.TMenuItem.Caption 1396
Menus.TMenuItem.Checked 1401
Menus.TMenuItem.Clear 1375
Menus.TMenuItem.Click 1375
Menus.TMenuItem.Command 1401
Menus.TMenuItem.Count 1402
Menus.TMenuItem.Create 1366
Menus.TMenuItem.Default 1406
Menus.TMenuItem.DefineProperties 1375
Menus.TMenuItem.Delete 1376
Menus.TMenuItem.Destroy 1367
Menus.TMenuItem.DoDrawText 1376
Menus.TMenuItem.DrawItem 1377
Menus.TMenuItem.Enabled 1406
Menus.TMenuItem.Find 1377
Menus.TMenuItem.GetActionLinkClass 1377
Menus.TMenuItem.GetAutoHotkeys 1378
Menus.TMenuItem.GetAutoLineReduction 1378

Menus.TMenuItem.GetChildren 1378
Menus.TMenuItem.GetCount 1378
Menus.TMenuItem.GetEnumerator 1379
Menus.TMenuItem.GetHandle 1379
Menus.TMenuItem.GetImageList 1379
Menus.TMenuItem.GetItem 1380
Menus.TMenuItem.GetMenuIndex 1380
Menus.TMenuItem.GetParentComponent 1380
Menus.TMenuItem.GetParentMenu 1380
Menus.TMenuItem.GroupIndex 1408
Menus.TMenuItem.Handle 1409
Menus.TMenuItem.HasParent 1381
Menus.TMenuItem.HelpContext 1410
Menus.TMenuItem.Hint 1410
Menus.TMenuItem.ImageIndex 1411
Menus.TMenuItem.IndexOf 1381
Menus.TMenuItem.InitiateAction 1382
Menus.TMenuItem.Insert 1382
Menus.TMenuItem.InsertNewLine 1383
Menus.TMenuItem.InsertNewLineAfter 1383
Menus.TMenuItem.InsertNewLineBefore 1384
Menus.TMenuItem.IsLine 1385
Menus.TMenuItem.Items 1411
Menus.TMenuItem.Loaded 1385
Menus.TMenuItem.MeasureItem 1386
Menus.TMenuItem.MenuChanged 1386
Menus.TMenuItem.MenuIndex 1413
Menus.TMenuItem.NewBottomLine 1386
Menus.TMenuItem.NewTopLine 1387
Menus.TMenuItem.Notification 1387
Menus.TMenuItem.OnAdvancedDrawItem 1416
Menus.TMenuItem.OnClick 1417
Menus.TMenuItem.OnDrawItem 1422
Menus.TMenuItem.OnMeasureItem 1423
Menus.TMenuItem.Parent 1414
Menus.TMenuItem.RadioItem 1414
Menus.TMenuItem.Remove 1388
Menus.TMenuItem.RethinkHotkeys 1388
Menus.TMenuItem.RethinkLines 1389
Menus.TMenuItem.SetBreak 1389
Menus.TMenuItem.SetCaption 1389
Menus.TMenuItem.SetChecked 1390
Menus.TMenuItem.SetChildOrder 1390
Menus.TMenuItem.SetDefault 1390
Menus.TMenuItem.SetEnabled 1390
Menus.TMenuItem.SetGroupIndex 1391
Menus.TMenuItem.SetImageIndex 1391
Menus.TMenuItem.SetMenuIndex 1391
Menus.TMenuItem.SetParentComponent 1391
Menus.TMenuItem.SetRadioItem 1392
Menus.TMenuItem.SetShortCut 1392
Menus.TMenuItem.SetVisible 1392
Menus.TMenuItem.ShortCut 1415
Menus.TMenuItem.SubMenuImages 1415
Menus.TMenuItem.UpdateItems 1392
Menus.TMenuItem.Visible 1416
Menus.TMenuItemAutoFlag 1462
Menus.TMenuItemAutoFlag enumeration 1462
Menus.TMenuItemEnumerator 1424
Menus.TMenuItemEnumerator.Create 1424
Menus.TMenuItemEnumerator.Current 1425
Menus.TMenuItemEnumerator.GetCurrent 1425
Menus.TMenuItemEnumerator.MoveNext 1425
Menus.TMenuItemStack 1426
Menus.TMenuItemStack.ClearItem 1426
Menus.TMenuMeasureItemEvent 1469
Menus.TMenuMeasureItemEvent type 1469
Menus.TPopupAlignment 1463
Menus.TPopupAlignment enumeration 1463
Menus.TPopupList 1427
Menus.TPopupList.Add 1428
Menus.TPopupList.FWindow 1428
Menus.TPopupList.MainWndProc 1429
Menus.TPopupList.Remove 1429
Menus.TPopupList.Window 1430
Menus.TPopupList.WndProc 1429
Menus.TPopupMenu 1430
Menus.TPopupMenu.Alignment 1436

Menus.TPopupMenu.AutoHotkeys 1436
Menus.TPopupMenu.AutoLineReduction 1437
Menus.TPopupMenu.AutoPopup 1437
Menus.TPopupMenu.BiDiMode 1438
Menus.TPopupMenu.Create 1432
Menus.TPopupMenu.Destroy 1433
Menus.TPopupMenu.DoPopup 1433
Menus.TPopupMenu.HelpContext 1438
Menus.TPopupMenu.Images 1439
Menus.TPopupMenu.MenuAnimation 1439
Menus.TPopupMenu.OnChange 1443
Menus.TPopupMenu.OnPopup 1444
Menus.TPopupMenu.OwnerDraw 1440
Menus.TPopupMenu.ParentBiDiMode 1440
Menus.TPopupMenu.Popup 1434
Menus.TPopupMenu.PopupComponent 1440
Menus.TPopupMenu.PopupPoint 1443
Menus.TPopupMenu.SetPopupPoint 1434
Menus.TPopupMenu.TrackButton 1443
Menus.TPopupMenu.UseRightToLeftAlignment 1435
Menus.TTrackButton 1463
Menus.TTrackButton enumeration 1463
Menus.ValidMenuHotkeys 1471
Menus.ValidMenuHotkeys variable 1471
Merge
TMainMenu class 1325
MessageFileLocation
TConfigParams class 403
Messages 1475
Messages namespace 1475
Constants 1595
Functions 1487
Structs, Records, Enums 1511
Types 1568
Messages.BM_CLICK 1602
Messages.BM_CLICK constant 1602
Messages.BM_GETCHECK 1603
Messages.BM_GETCHECK constant 1603
Messages.BM_GETIMAGE 1603
Messages.BM_GETIMAGE constant 1603
Messages.BM_GETSTATE 1604
Messages.BM_GETSTATE constant 1604
Messages.BM_SETCHECK 1604
Messages.BM_SETCHECK constant 1604
Messages.BM_SETIMAGE 1604
Messages.BM_SETIMAGE constant 1604
Messages.BM_SETSTATE 1605
Messages.BM_SETSTATE constant 1605
Messages.BM_SETSTYLE 1605
Messages.BM_SETSTYLE constant 1605
Messages.BN_CLICKED 1606
Messages.BN_CLICKED constant 1606
Messages.BN_DBLCLK 1606
Messages.BN_DBLCLK constant 1606
Messages.BN_DISABLE 1607
Messages.BN_DISABLE constant 1607
Messages.BN_DOUBLECLICKED 1607
Messages.BN_DOUBLECLICKED constant 1607
Messages.BN_HILITE 1607
Messages.BN_HILITE constant 1607
Messages.BN_KILLFOCUS 1608
Messages.BN_KILLFOCUS constant 1608
Messages.BN_PAINT 1608
Messages.BN_PAINT constant 1608
Messages.BN_PUSHED 1609
Messages.BN_PUSHED constant 1609
Messages.BN_SETFOCUS 1609
Messages.BN_SETFOCUS constant 1609
Messages.BN_UNHILITE 1610
Messages.BN_UNHILITE constant 1610
Messages.BN_UNPUSHED 1610
Messages.BN_UNPUSHED constant 1610
Messages.CB_ADDSTRING 1610
Messages.CB_ADDSTRING constant 1610
Messages.CB_DELETESTRING 1611
Messages.CB_DELETESTRING constant 1611
Messages.CB_DIR 1611
Messages.CB_DIR constant 1611

Messages.CB_FINDSTRING 1612
Messages.CB_FINDSTRING constant 1612
Messages.CB_FINDSTRINGEXACT 1612
Messages.CB_FINDSTRINGEXACT constant 1612
Messages.CB_GETCOUNT 1613
Messages.CB_GETCOUNT constant 1613
Messages.CB_GETCURSEL 1613
Messages.CB_GETCURSEL constant 1613
Messages.CB_GETDROPPEDCONTROLRECT 1613
Messages.CB_GETDROPPEDCONTROLRECT constant 1613
Messages.CB_GETDROPPEDSTATE 1614
Messages.CB_GETDROPPEDSTATE constant 1614
Messages.CB_GETDROPPEDWIDTH 1614
Messages.CB_GETDROPPEDWIDTH constant 1614
Messages.CB_GETEDITSEL 1615
Messages.CB_GETEDITSEL constant 1615
Messages.CB_GETEXTENDEDUI 1615
Messages.CB_GETEXTENDEDUI constant 1615
Messages.CB_GETHORIZONTALEXTENT 1616
Messages.CB_GETHORIZONTALEXTENT constant 1616
Messages.CB_GETITEMDATA 1616
Messages.CB_GETITEMDATA constant 1616
Messages.CB_GETITEMHEIGHT 1617
Messages.CB_GETITEMHEIGHT constant 1617
Messages.CB_GETLBTEXT 1617
Messages.CB_GETLBTEXT constant 1617
Messages.CB_GETLBTEXTLEN 1617
Messages.CB_GETLBTEXTLEN constant 1617
Messages.CB_GETLOCALE 1618
Messages.CB_GETLOCALE constant 1618
Messages.CB_GETTOPINDEX 1618
Messages.CB_GETTOPINDEX constant 1618
Messages.CB_INITSTORAGE 1619
Messages.CB_INITSTORAGE constant 1619
Messages.CB_INSERTSTRING 1619
Messages.CB_INSERTSTRING constant 1619
Messages.CB_LIMITTEXT 1620
Messages.CB_LIMITTEXT constant 1620
Messages.CB_MSGMAX 1620
Messages.CB_MSGMAX constant 1620
Messages.CB_RESETCONTENT 1620
Messages.CB_RESETCONTENT constant 1620
Messages.CB_SELECTSTRING 1621
Messages.CB_SELECTSTRING constant 1621
Messages.CB_SETCURSEL 1621
Messages.CB_SETCURSEL constant 1621
Messages.CB_SETDROPPEDWIDTH 1622
Messages.CB_SETDROPPEDWIDTH constant 1622
Messages.CB_SETEDITSEL 1622
Messages.CB_SETEDITSEL constant 1622
Messages.CB_SETEXTENDEDUI 1623
Messages.CB_SETEXTENDEDUI constant 1623
Messages.CB_SETHORIZONTALEXTENT 1623
Messages.CB_SETHORIZONTALEXTENT constant 1623
Messages.CB_SETITEMDATA 1623
Messages.CB_SETITEMDATA constant 1623
Messages.CB_SETITEMHEIGHT 1624
Messages.CB_SETITEMHEIGHT constant 1624
Messages.CB_SETLOCALE 1624
Messages.CB_SETLOCALE constant 1624
Messages.CB_SETTOPINDEX 1625
Messages.CB_SETTOPINDEX constant 1625
Messages.CB_SHOWDROPDOWN 1625
Messages.CB_SHOWDROPDOWN constant 1625
Messages.CBN_CLOSEUP 1626
Messages.CBN_CLOSEUP constant 1626
Messages.CBN_DBLCLK 1626
Messages.CBN_DBLCLK constant 1626
Messages.CBN_DROPDOWN 1626
Messages.CBN_DROPDOWN constant 1626
Messages.CBN_EDITCHANGE 1627
Messages.CBN_EDITCHANGE constant 1627
Messages.CBN_EDITUPDATE 1627
Messages.CBN_EDITUPDATE constant 1627
Messages.CBN_ERRSPACE 1628
Messages.CBN_ERRSPACE constant 1628
Messages.CBN_KILLFOCUS 1628
Messages.CBN_KILLFOCUS constant 1628

Messages.CBN_SELCHANGE 1629
Messages.CBN_SELCHANGE constant 1629
Messages.CBN_SELENDCANCEL 1629
Messages.CBN_SELENDCANCEL constant 1629
Messages.CBN_SELENDOK 1629
Messages.CBN_SELENDOK constant 1629
Messages.CBN_SETFOCUS 1630
Messages.CBN_SETFOCUS constant 1630
Messages.DM_GETDEFID 1630
Messages.DM_GETDEFID constant 1630
Messages.DM_REPOSITION 1631
Messages.DM_REPOSITION constant 1631
Messages.DM_SETDEFID 1631
Messages.DM_SETDEFID constant 1631
Messages.EM_CANUNDO 1632
Messages.EM_CANUNDO constant 1632
Messages.EM_CHARFROMPOS 1632
Messages.EM_CHARFROMPOS constant 1632
Messages.EM_EMPTYUNDOBUFFER 1632
Messages.EM_EMPTYUNDOBUFFER constant 1632
Messages.EM_FMTLINES 1633
Messages.EM_FMTLINES constant 1633
Messages.EM_GETFIRSTVISIBLELINE 1633
Messages.EM_GETFIRSTVISIBLELINE constant 1633
Messages.EM_GETHANDLE 1634
Messages.EM_GETHANDLE constant 1634
Messages.EM_GETIMESTATUS 1634
Messages.EM_GETIMESTATUS constant 1634
Messages.EM_GETLIMITTEXT 1635
Messages.EM_GETLIMITTEXT constant 1635
Messages.EM_GETLINE 1635
Messages.EM_GETLINE constant 1635
Messages.EM_GETLINECOUNT 1635
Messages.EM_GETLINECOUNT constant 1635
Messages.EM_GETMARGINS 1636
Messages.EM_GETMARGINS constant 1636
Messages.EM_GETMODIFY 1636
Messages.EM_GETMODIFY constant 1636
Messages.EM_GETPASSWORDCHAR 1637
Messages.EM_GETPASSWORDCHAR constant 1637
Messages.EM_GETRECT 1637
Messages.EM_GETRECT constant 1637
Messages.EM_GETSEL 1638
Messages.EM_GETSEL constant 1638
Messages.EM_GETTHUMB 1638
Messages.EM_GETTHUMB constant 1638
Messages.EM_GETWORDBREAKPROC 1638
Messages.EM_GETWORDBREAKPROC constant 1638
Messages.EM_LIMITTEXT 1639
Messages.EM_LIMITTEXT constant 1639
Messages.EM_LINEFROMCHAR 1639
Messages.EM_LINEFROMCHAR constant 1639
Messages.EM_LINEINDEX 1640
Messages.EM_LINEINDEX constant 1640
Messages.EM_LINELENGTH 1640
Messages.EM_LINELENGTH constant 1640
Messages.EM_LINESCROLL 1641
Messages.EM_LINESCROLL constant 1641
Messages.EM_POSFROMCHAR 1641
Messages.EM_POSFROMCHAR constant 1641
Messages.EM_REPLACESEL 1641
Messages.EM_REPLACESEL constant 1641
Messages.EM_SCROLL 1642
Messages.EM_SCROLL constant 1642
Messages.EM_SCROLLCARET 1642
Messages.EM_SCROLLCARET constant 1642
Messages.EM_SETHANDLE 1643
Messages.EM_SETHANDLE constant 1643
Messages.EM_SETIMESTATUS 1643
Messages.EM_SETIMESTATUS constant 1643
Messages.EM_SETLIMITTEXT 1644
Messages.EM_SETLIMITTEXT constant 1644
Messages.EM_SETMARGINS 1644
Messages.EM_SETMARGINS constant 1644
Messages.EM_SETMODIFY 1644
Messages.EM_SETMODIFY constant 1644
Messages.EM_SETPASSWORDCHAR 1645
Messages.EM_SETPASSWORDCHAR constant 1645

Messages.EM_SETREADONLY 1645
Messages.EM_SETREADONLY constant 1645
Messages.EM_SETRECT 1646
Messages.EM_SETRECT constant 1646
Messages.EM_SETRECTNP 1646
Messages.EM_SETRECTNP constant 1646
Messages.EM_SETSEL 1647
Messages.EM_SETSEL constant 1647
Messages.EM_SETTABSTOPS 1647
Messages.EM_SETTABSTOPS constant 1647
Messages.EM_SETWORDBREAKPROC 1647
Messages.EM_SETWORDBREAKPROC constant 1647
Messages.EM_UNDO 1648
Messages.EM_UNDO constant 1648
Messages.EN_CHANGE 1648
Messages.EN_CHANGE constant 1648
Messages.EN_ERRSPACE 1649
Messages.EN_ERRSPACE constant 1649
Messages.EN_HSCROLL 1649
Messages.EN_HSCROLL constant 1649
Messages.EN_KILLFOCUS 1650
Messages.EN_KILLFOCUS constant 1650
Messages.EN_MAXTEXT 1650
Messages.EN_MAXTEXT constant 1650
Messages.EN_SETFOCUS 1650
Messages.EN_SETFOCUS constant 1650
Messages.EN_UPDATE 1651
Messages.EN_UPDATE constant 1651
Messages.EN_VSCROLL 1651
Messages.EN_VSCROLL constant 1651
Messages.LB_ADDFILE 1652
Messages.LB_ADDFILE constant 1652
Messages.LB_ADDSTRING 1652
Messages.LB_ADDSTRING constant 1652
Messages.LB_DELETESTRING 1653
Messages.LB_DELETESTRING constant 1653
Messages.LB_DIR 1653
Messages.LB_DIR constant 1653
Messages.LB_FINDSTRING 1653
Messages.LB_FINDSTRING constant 1653
Messages.LB_FINDSTRINGEXACT 1654
Messages.LB_FINDSTRINGEXACT constant 1654
Messages.LB_GETANCHORINDEX 1654
Messages.LB_GETANCHORINDEX constant 1654
Messages.LB_GETCARETINDEX 1655
Messages.LB_GETCARETINDEX constant 1655
Messages.LB_GETCOUNT 1655
Messages.LB_GETCOUNT constant 1655
Messages.LB_GETCURSEL 1656
Messages.LB_GETCURSEL constant 1656
Messages.LB_GETHORIZONTALEXTENT 1656
Messages.LB_GETHORIZONTALEXTENT constant 1656
Messages.LB_GETITEMDATA 1656
Messages.LB_GETITEMDATA constant 1656
Messages.LB_GETITEMHEIGHT 1657
Messages.LB_GETITEMHEIGHT constant 1657
Messages.LB_GETITEMRECT 1657
Messages.LB_GETITEMRECT constant 1657
Messages.LB_GETLOCALE 1658
Messages.LB_GETLOCALE constant 1658
Messages.LB_GETSEL 1658
Messages.LB_GETSEL constant 1658
Messages.LB_GETSELCOUNT 1659
Messages.LB_GETSELCOUNT constant 1659
Messages.LB_GETSELITEMS 1659
Messages.LB_GETSELITEMS constant 1659
Messages.LB_GETTEXT 1659
Messages.LB_GETTEXT constant 1659
Messages.LB_GETTEXTLEN 1660
Messages.LB_GETTEXTLEN constant 1660
Messages.LB_GETTOPINDEX 1660
Messages.LB_GETTOPINDEX constant 1660
Messages.LB_INITSTORAGE 1661
Messages.LB_INITSTORAGE constant 1661
Messages.LB_INSERTSTRING 1661
Messages.LB_INSERTSTRING constant 1661
Messages.LB_ITEMFROMPOINT 1662
Messages.LB_ITEMFROMPOINT constant 1662

Messages.LB_MSGMAX 1662
Messages.LB_MSGMAX constant 1662
Messages.LB_RESETCONTENT 1662
Messages.LB_RESETCONTENT constant 1662
Messages.LB_SELECTSTRING 1663
Messages.LB_SELECTSTRING constant 1663
Messages.LB_SELITEMRANGE 1663
Messages.LB_SELITEMRANGE constant 1663
Messages.LB_SELITEMRANGEEX 1664
Messages.LB_SELITEMRANGEEX constant 1664
Messages.LB_SETANCHORINDEX 1664
Messages.LB_SETANCHORINDEX constant 1664
Messages.LB_SETCARETINDEX 1665
Messages.LB_SETCARETINDEX constant 1665
Messages.LB_SETCOLUMNWIDTH 1665
Messages.LB_SETCOLUMNWIDTH constant 1665
Messages.LB_SETCOUNT 1665
Messages.LB_SETCOUNT constant 1665
Messages.LB_SETCURSEL 1666
Messages.LB_SETCURSEL constant 1666
Messages.LB_SETHORIZONTALEXTENT 1666
Messages.LB_SETHORIZONTALEXTENT constant 1666
Messages.LB_SETITEMDATA 1667
Messages.LB_SETITEMDATA constant 1667
Messages.LB_SETITEMHEIGHT 1667
Messages.LB_SETITEMHEIGHT constant 1667
Messages.LB_SETLOCALE 1668
Messages.LB_SETLOCALE constant 1668
Messages.LB_SETSEL 1668
Messages.LB_SETSEL constant 1668
Messages.LB_SETTABSTOPS 1668
Messages.LB_SETTABSTOPS constant 1668
Messages.LB_SETTOPINDEX 1669
Messages.LB_SETTOPINDEX constant 1669
Messages.LBN_DBLCLK 1669
Messages.LBN_DBLCLK constant 1669
Messages.LBN_ERRSPACE 1670
Messages.LBN_ERRSPACE constant 1670
Messages.LBN_KILLFOCUS 1670
Messages.LBN_KILLFOCUS constant 1670
Messages.LBN_SELCANCEL 1671
Messages.LBN_SELCANCEL constant 1671
Messages.LBN_SELCHANGE 1671
Messages.LBN_SELCHANGE constant 1671
Messages.LBN_SETFOCUS 1671
Messages.LBN_SETFOCUS constant 1671
Messages.PMessage 1569
Messages.PMessage type 1569
Messages.PSM_PAGEINFO 1672
Messages.PSM_PAGEINFO constant 1672
Messages.PSM_SHEETINFO 1672
Messages.PSM_SHEETINFO constant 1672
Messages.SBM_ENABLE_ARROWS 1673
Messages.SBM_ENABLE_ARROWS constant 1673
Messages.SBM_GETPOS 1673
Messages.SBM_GETPOS constant 1673
Messages.SBM_GETRANGE 1674
Messages.SBM_GETRANGE constant 1674
Messages.SBM_GETSCROLLBARINFO 1674
Messages.SBM_GETSCROLLBARINFO constant 1674
Messages.SBM_GETSCROLLINFO 1674
Messages.SBM_GETSCROLLINFO constant 1674
Messages.SBM_SETPOS 1675
Messages.SBM_SETPOS constant 1675
Messages.SBM_SETRANGE 1675
Messages.SBM_SETRANGE constant 1675
Messages.SBM_SETRANGEREDRAW 1676
Messages.SBM_SETRANGEREDRAW constant 1676
Messages.SBM_SETSCROLLINFO 1676
Messages.SBM_SETSCROLLINFO constant 1676
Messages.SendGetIntMessage 1487
Messages.SendGetIntMessage function 1487
Messages.SendGetStructMessage 1493
Messages.SendGetStructMessage function 1493
Messages.SendStructMessage 1499
Messages.SendStructMessage function 1499
Messages.SendTextMessage 1505
Messages.SendTextMessage function 1505

Messages.TMessage 1513
Messages.TMessage record 1513
Messages.TMSHMouseWheel 1514
Messages.TMSHMouseWheel record 1514
Messages.TWMActivate 1514
Messages.TWMActivate record 1514
Messages.TWMActivateApp 1515
Messages.TWMActivateApp record 1515
Messages.TWMAskCBFormatName 1515
Messages.TWMAskCBFormatName record 1515
Messages.TWMCancelMode 1570
Messages.TWMCancelMode type 1570
Messages.TWMChangeCBChain 1516
Messages.TWMChangeCBChain record 1516
Messages.TWMChangeUIState 1570
Messages.TWMChangeUIState type 1570
Messages.TWMChar 1570
Messages.TWMChar type 1570
Messages.TWMCharToItem 1516
Messages.TWMCharToItem record 1516
Messages.TWMChildActivate 1571
Messages.TWMChildActivate type 1571
Messages.TWMChooseFont_GetLogFont 1517
Messages.TWMChooseFont_GetLogFont record 1517
Messages.TWMClear 1571
Messages.TWMClear type 1571
Messages.TWMClose 1571
Messages.TWMClose type 1571
Messages.TWMCommand 1517
Messages.TWMCommand record 1517
Messages.TWMCompacting 1518
Messages.TWMCompacting record 1518
Messages.TWMCompareItem 1518
Messages.TWMCompareItem record 1518
Messages.TWMContextMenu 1519
Messages.TWMContextMenu record 1519
Messages.TWMCopy 1572
Messages.TWMCopy type 1572
Messages.TWMCopyData 1519
Messages.TWMCopyData record 1519
Messages.TWMCreate 1520
Messages.TWMCreate record 1520
Messages.TWMCtlColor 1520
Messages.TWMCtlColor record 1520
Messages.TWMCtlColorBtn 1572
Messages.TWMCtlColorBtn type 1572
Messages.TWMCtlColorDlg 1572
Messages.TWMCtlColorDlg type 1572
Messages.TWMCtlColorEdit 1573
Messages.TWMCtlColorEdit type 1573
Messages.TWMCtlColorListbox 1573
Messages.TWMCtlColorListbox type 1573
Messages.TWMCtlColorMsgbox 1573
Messages.TWMCtlColorMsgbox type 1573
Messages.TWMCtlColorScrollbar 1574
Messages.TWMCtlColorScrollbar type 1574
Messages.TWMCtlColorStatic 1574
Messages.TWMCtlColorStatic type 1574
Messages.TWMCut 1574
Messages.TWMCut type 1574
Messages.TWMDDE_Ack 1521
Messages.TWMDDE_Ack record 1521
Messages.TWMDDE_Advise 1521
Messages.TWMDDE_Advise record 1521
Messages.TWMDDE_Data 1522
Messages.TWMDDE_Data record 1522
Messages.TWMDDE_Execute 1522
Messages.TWMDDE_Execute record 1522
Messages.TWMDDE_Initiate 1523
Messages.TWMDDE_Initiate record 1523
Messages.TWMDDE_Poke 1523
Messages.TWMDDE_Poke record 1523
Messages.TWMDDE_Request 1524
Messages.TWMDDE_Request record 1524
Messages.TWMDDE_Terminate 1524
Messages.TWMDDE_Terminate record 1524
Messages.TWMDDE_Unadvise 1525
Messages.TWMDDE_Unadvise record 1525

Messages.TWMDeadChar 1575
Messages.TWMDeadChar type 1575
Messages.TWMDeleteItem 1525
Messages.TWMDeleteItem record 1525
Messages.TWMDestroy 1575
Messages.TWMDestroy type 1575
Messages.TWMDestroyClipboard 1575
Messages.TWMDestroyClipboard type 1575
Messages.TWMDevModeChange 1526
Messages.TWMDevModeChange record 1526
Messages.TWMDisplayChange 1526
Messages.TWMDisplayChange record 1526
Messages.TWMDrawClipboard 1576
Messages.TWMDrawClipboard type 1576
Messages.TWMDrawItem 1527
Messages.TWMDrawItem record 1527
Messages.TWMDropFiles 1527
Messages.TWMDropFiles record 1527
Messages.TWMEnable 1528
Messages.TWMEnable record 1528
Messages.TWMEndSession 1528
Messages.TWMEndSession record 1528
Messages.TWMEnterIdle 1529
Messages.TWMEnterIdle record 1529
Messages.TWMEnterMenuLoop 1529
Messages.TWMEnterMenuLoop record 1529
Messages.TWMEraseBkgnd 1530
Messages.TWMEraseBkgnd record 1530
Messages.TWMExitMenuLoop 1576
Messages.TWMExitMenuLoop type 1576
Messages.TWMFontChange 1576
Messages.TWMFontChange type 1576
Messages.TWMGetDlgCode 1577
Messages.TWMGetDlgCode type 1577
Messages.TWMGetFont 1577
Messages.TWMGetFont type 1577
Messages.TWMGetHotKey 1577
Messages.TWMGetHotKey type 1577
Messages.TWMGetIcon 1530
Messages.TWMGetIcon record 1530
Messages.TWMGetMinMaxInfo 1531
Messages.TWMGetMinMaxInfo record 1531
Messages.TWMGetText 1531
Messages.TWMGetText record 1531
Messages.TWMGetTextLength 1578
Messages.TWMGetTextLength type 1578
Messages.TWMHelp 1532
Messages.TWMHelp record 1532
Messages.TWMHotKey 1532
Messages.TWMHotKey record 1532
Messages.TWMHScroll 1578
Messages.TWMHScroll type 1578
Messages.TWMHScrollClipboard 1533
Messages.TWMHScrollClipboard record 1533
Messages.TWMIconEraseBkgnd 1578
Messages.TWMIconEraseBkgnd type 1578
Messages.TWMInitDialog 1533
Messages.TWMInitDialog record 1533
Messages.TWMInitMenu 1534
Messages.TWMInitMenu record 1534
Messages.TWMInitMenuPopup 1534
Messages.TWMInitMenuPopup record 1534
Messages.TWMKey 1535
Messages.TWMKey record 1535
Messages.TWMKeyDown 1579
Messages.TWMKeyDown type 1579
Messages.TWMKeyUp 1579
Messages.TWMKeyUp type 1579
Messages.TWMKillFocus 1536
Messages.TWMKillFocus record 1536
Messages.TWMLButtonDblClk 1579
Messages.TWMLButtonDblClk type 1579
Messages.TWMLButtonDown 1580
Messages.TWMLButtonDown type 1580
Messages.TWMLButtonUp 1580
Messages.TWMLButtonUp type 1580
Messages.TWMMButtonDblClk 1580
Messages.TWMMButtonDblClk type 1580

Messages.TWMMButtonDown 1581
Messages.TWMMButtonDown type 1581
Messages.TWMMButtonUp 1581
Messages.TWMMButtonUp type 1581
Messages.TWMMDIActivate 1536
Messages.TWMMDIActivate record 1536
Messages.TWMMDICascade 1537
Messages.TWMMDICascade record 1537
Messages.TWMMDICreate 1537
Messages.TWMMDICreate record 1537
Messages.TWMMDIDestroy 1538
Messages.TWMMDIDestroy record 1538
Messages.TWMMDIGetActive 1581
Messages.TWMMDIGetActive type 1581
Messages.TWMMDIIconArrange 1582
Messages.TWMMDIIconArrange type 1582
Messages.TWMMDIMaximize 1538
Messages.TWMMDIMaximize record 1538
Messages.TWMMDINext 1539
Messages.TWMMDINext record 1539
Messages.TWMMDIRefreshMenu 1582
Messages.TWMMDIRefreshMenu type 1582
Messages.TWMMDIRestore 1539
Messages.TWMMDIRestore record 1539
Messages.TWMMDISetMenu 1540
Messages.TWMMDISetMenu record 1540
Messages.TWMMDITile 1540
Messages.TWMMDITile record 1540
Messages.TWMMeasureItem 1541
Messages.TWMMeasureItem record 1541
Messages.TWMMenuChar 1541
Messages.TWMMenuChar record 1541
Messages.TWMMenuSelect 1542
Messages.TWMMenuSelect record 1542
Messages.TWMMouse 1543
Messages.TWMMouse record 1543
Messages.TWMMouseActivate 1544
Messages.TWMMouseActivate record 1544
Messages.TWMMouseMove 1582
Messages.TWMMouseMove type 1582
Messages.TWMMouseWheel 1544
Messages.TWMMouseWheel record 1544
Messages.TWMMove 1545
Messages.TWMMove record 1545
Messages.TWMMoving 1545
Messages.TWMMoving record 1545
Messages.TWMNCActivate 1546
Messages.TWMNCActivate record 1546
Messages.TWMNCCalcSize 1546
Messages.TWMNCCalcSize record 1546
Messages.TWMNCCreate 1547
Messages.TWMNCCreate record 1547
Messages.TWMNCDestroy 1583
Messages.TWMNCDestroy type 1583
Messages.TWMNCHitMessage 1547
Messages.TWMNCHitMessage record 1547
Messages.TWMNCHitTest 1548
Messages.TWMNCHitTest record 1548
Messages.TWMNCLButtonDblClk 1583
Messages.TWMNCLButtonDblClk type 1583
Messages.TWMNCLButtonDown 1583
Messages.TWMNCLButtonDown type 1583
Messages.TWMNCLButtonUp 1584
Messages.TWMNCLButtonUp type 1584
Messages.TWMNCMButtonDblClk 1584
Messages.TWMNCMButtonDblClk type 1584
Messages.TWMNCMButtonDown 1584
Messages.TWMNCMButtonDown type 1584
Messages.TWMNCMButtonUp 1585
Messages.TWMNCMButtonUp type 1585
Messages.TWMNCMouseMove 1585
Messages.TWMNCMouseMove type 1585
Messages.TWMNCPaint 1548
Messages.TWMNCPaint record 1548
Messages.TWMNCRButtonDblClk 1585
Messages.TWMNCRButtonDblClk type 1585
Messages.TWMNCRButtonDown 1586
Messages.TWMNCRButtonDown type 1586

Messages.TWMNCRButtonUp 1586
Messages.TWMNCRButtonUp type 1586
Messages.TWMNextDlgCtl 1549
Messages.TWMNextDlgCtl record 1549
Messages.TWMNoParams 1549
Messages.TWMNoParams record 1549
Messages.TWMNotify 1550
Messages.TWMNotify record 1550
Messages.TWMNotifyFormat 1550
Messages.TWMNotifyFormat record 1550
Messages.TWMPaint 1551
Messages.TWMPaint record 1551
Messages.TWMPaintClipboard 1551
Messages.TWMPaintClipboard record 1551
Messages.TWMPaintIcon 1586
Messages.TWMPaintIcon type 1586
Messages.TWMPaletteChanged 1552
Messages.TWMPaletteChanged record 1552
Messages.TWMPaletteIsChanging 1552
Messages.TWMPaletteIsChanging record 1552
Messages.TWMParentNotify 1553
Messages.TWMParentNotify record 1553
Messages.TWMPaste 1587
Messages.TWMPaste type 1587
Messages.TWMPower 1554
Messages.TWMPower record 1554
Messages.TWMPrint 1554
Messages.TWMPrint record 1554
Messages.TWMPrintClient 1587
Messages.TWMPrintClient type 1587
Messages.TWMQueryDragIcon 1587
Messages.TWMQueryDragIcon type 1587
Messages.TWMQueryEndSession 1555
Messages.TWMQueryEndSession record 1555
Messages.TWMQueryNewPalette 1588
Messages.TWMQueryNewPalette type 1588
Messages.TWMQueryOpen 1588
Messages.TWMQueryOpen type 1588
Messages.TWMQueryUIState 1588
Messages.TWMQueryUIState type 1588
Messages.TWMQueueSync 1589
Messages.TWMQueueSync type 1589
Messages.TWMQuit 1555
Messages.TWMQuit record 1555
Messages.TWMRButtonDblClk 1589
Messages.TWMRButtonDblClk type 1589
Messages.TWMRButtonDown 1589
Messages.TWMRButtonDown type 1589
Messages.TWMRButtonUp 1590
Messages.TWMRButtonUp type 1590
Messages.TWMRenderAllFormats 1590
Messages.TWMRenderAllFormats type 1590
Messages.TWMRenderFormat 1556
Messages.TWMRenderFormat record 1556
Messages.TWMScroll 1556
Messages.TWMScroll record 1556
Messages.TWMSetCursor 1557
Messages.TWMSetCursor record 1557
Messages.TWMSetFocus 1557
Messages.TWMSetFocus record 1557
Messages.TWMSetFont 1558
Messages.TWMSetFont record 1558
Messages.TWMSetHotKey 1558
Messages.TWMSetHotKey record 1558
Messages.TWMSetIcon 1559
Messages.TWMSetIcon record 1559
Messages.TWMSetRedraw 1559
Messages.TWMSetRedraw record 1559
Messages.TWMSetText 1560
Messages.TWMSetText record 1560
Messages.TWMSettingChange 1560
Messages.TWMSettingChange record 1560
Messages.TWMShowWindow 1561
Messages.TWMShowWindow record 1561
Messages.TWMSize 1561
Messages.TWMSize record 1561
Messages.TWMSizeClipboard 1562
Messages.TWMSizeClipboard record 1562

Messages.TWMSpoolerStatus 1562
Messages.TWMSpoolerStatus record 1562
Messages.TWMStyleChange 1563
Messages.TWMStyleChange record 1563
Messages.TWMStyleChanged 1590
Messages.TWMStyleChanged type 1590
Messages.TWMStyleChanging 1591
Messages.TWMStyleChanging type 1591
Messages.TWMSysChar 1591
Messages.TWMSysChar type 1591
Messages.TWMSysColorChange 1591
Messages.TWMSysColorChange type 1591
Messages.TWMSysCommand 1563
Messages.TWMSysCommand record 1563
Messages.TWMSysDeadChar 1564
Messages.TWMSysDeadChar record 1564
Messages.TWMSysKeyDown 1592
Messages.TWMSysKeyDown type 1592
Messages.TWMSysKeyUp 1592
Messages.TWMSysKeyUp type 1592
Messages.TWMSystemError 1564
Messages.TWMSystemError record 1564
Messages.TWMTimeChange 1592
Messages.TWMTimeChange type 1592
Messages.TWMTimer 1565
Messages.TWMTimer record 1565
Messages.TWMUIState 1565
Messages.TWMUIState record 1565
Messages.TWMUndo 1593
Messages.TWMUndo type 1593
Messages.TWMUpdateUIState 1593
Messages.TWMUpdateUIState type 1593
Messages.TWMVKeyToItem 1593
Messages.TWMVKeyToItem type 1593
Messages.TWMVScroll 1594
Messages.TWMVScroll type 1594
Messages.TWMVScrollClipboard 1566
Messages.TWMVScrollClipboard record 1566
Messages.TWMWindowPosChanged 1594
Messages.TWMWindowPosChanged type 1594
Messages.TWMWindowPosChanging 1594
Messages.TWMWindowPosChanging type 1594
Messages.TWMWindowPosMsg 1566
Messages.TWMWindowPosMsg record 1566
Messages.TWMWinIniChange 1567
Messages.TWMWinIniChange record 1567
Messages.WM_ACTIVATE 1677
Messages.WM_ACTIVATE constant 1677
Messages.WM_ACTIVATEAPP 1677
Messages.WM_ACTIVATEAPP constant 1677
Messages.WM_APP 1678
Messages.WM_APP constant 1678
Messages.WM_APPCOMMAND 1678
Messages.WM_APPCOMMAND constant 1678
Messages.WM_ASKCBFORMATNAME 1678
Messages.WM_ASKCBFORMATNAME constant 1678
Messages.WM_CANCELJOURNAL 1679
Messages.WM_CANCELJOURNAL constant 1679
Messages.WM_CANCELMODE 1679
Messages.WM_CANCELMODE constant 1679
Messages.WM_CAPTURECHANGED 1680
Messages.WM_CAPTURECHANGED constant 1680
Messages.WM_CHANGECBCHAIN 1680
Messages.WM_CHANGECBCHAIN constant 1680
Messages.WM_CHANGEUISTATE 1681
Messages.WM_CHANGEUISTATE constant 1681
Messages.WM_CHAR 1681
Messages.WM_CHAR constant 1681
Messages.WM_CHARTOITEM 1681
Messages.WM_CHARTOITEM constant 1681
Messages.WM_CHILDACTIVATE 1682
Messages.WM_CHILDACTIVATE constant 1682
Messages.WM_CLEAR 1682
Messages.WM_CLEAR constant 1682
Messages.WM_CLOSE 1683
Messages.WM_CLOSE constant 1683
Messages.WM_COALESCE_FIRST 1683
Messages.WM_COALESCE_FIRST constant 1683

Messages.WM_COALESCE_LAST 1684
Messages.WM_COALESCE_LAST constant 1684
Messages.WM_COMMAND 1684
Messages.WM_COMMAND constant 1684
Messages.WM_COMMNOTIFY 1685
Messages.WM_COMMNOTIFY constant 1685
Messages.WM_COMPACTING 1685
Messages.WM_COMPACTING constant 1685
Messages.WM_COMPAREITEM 1685
Messages.WM_COMPAREITEM constant 1685
Messages.WM_CONTEXTMENU 1686
Messages.WM_CONTEXTMENU constant 1686
Messages.WM_COPY 1686
Messages.WM_COPY constant 1686
Messages.WM_COPYDATA 1687
Messages.WM_COPYDATA constant 1687
Messages.WM_CREATE 1687
Messages.WM_CREATE constant 1687
Messages.WM_CTLCOLOR 1688
Messages.WM_CTLCOLOR constant 1688
Messages.WM_CTLCOLORBTN 1688
Messages.WM_CTLCOLORBTN constant 1688
Messages.WM_CTLCOLORDLG 1688
Messages.WM_CTLCOLORDLG constant 1688
Messages.WM_CTLCOLOREDIT 1689
Messages.WM_CTLCOLOREDIT constant 1689
Messages.WM_CTLCOLORLISTBOX 1689
Messages.WM_CTLCOLORLISTBOX constant 1689
Messages.WM_CTLCOLORMSGBOX 1690
Messages.WM_CTLCOLORMSGBOX constant 1690
Messages.WM_CTLCOLORSCROLLBAR 1690
Messages.WM_CTLCOLORSCROLLBAR constant 1690
Messages.WM_CTLCOLORSTATIC 1691
Messages.WM_CTLCOLORSTATIC constant 1691
Messages.WM_CUT 1691
Messages.WM_CUT constant 1691
Messages.WM_DDE_ACK 1691
Messages.WM_DDE_ACK constant 1691
Messages.WM_DDE_ADVISE 1692
Messages.WM_DDE_ADVISE constant 1692
Messages.WM_DDE_DATA 1692
Messages.WM_DDE_DATA constant 1692
Messages.WM_DDE_EXECUTE 1693
Messages.WM_DDE_EXECUTE constant 1693
Messages.WM_DDE_FIRST 1693
Messages.WM_DDE_FIRST constant 1693
Messages.WM_DDE_INITIATE 1694
Messages.WM_DDE_INITIATE constant 1694
Messages.WM_DDE_LAST 1694
Messages.WM_DDE_LAST constant 1694
Messages.WM_DDE_POKE 1695
Messages.WM_DDE_POKE constant 1695
Messages.WM_DDE_REQUEST 1695
Messages.WM_DDE_REQUEST constant 1695
Messages.WM_DDE_TERMINATE 1695
Messages.WM_DDE_TERMINATE constant 1695
Messages.WM_DDE_UNADVISE 1696
Messages.WM_DDE_UNADVISE constant 1696
Messages.WM_DEADCHAR 1696
Messages.WM_DEADCHAR constant 1696
Messages.WM_DELETEITEM 1697
Messages.WM_DELETEITEM constant 1697
Messages.WM_DESTROY 1697
Messages.WM_DESTROY constant 1697
Messages.WM_DESTROYCLIPBOARD 1698
Messages.WM_DESTROYCLIPBOARD constant 1698
Messages.WM_DEVICECHANGE 1698
Messages.WM_DEVICECHANGE constant 1698
Messages.WM_DEVMODECHANGE 1698
Messages.WM_DEVMODECHANGE constant 1698
Messages.WM_DISPLAYCHANGE 1699
Messages.WM_DISPLAYCHANGE constant 1699
Messages.WM_DRAWCLIPBOARD 1699
Messages.WM_DRAWCLIPBOARD constant 1699
Messages.WM_DRAWITEM 1700
Messages.WM_DRAWITEM constant 1700
Messages.WM_DROPFILES 1700
Messages.WM_DROPFILES constant 1700

Messages.WM_DWMCOLORIZATIONCOLORCHANGED 1701
Messages.WM_DWMCOLORIZATIONCOLORCHANGED constant 1701
Messages.WM_DWMCOMPOSITIONCHANGED 1701
Messages.WM_DWMCOMPOSITIONCHANGED constant 1701
Messages.WM_DWMNCRENDERINGCHANGED 1702
Messages.WM_DWMNCRENDERINGCHANGED constant 1702
Messages.WM_DWMWINDOWMAXIMIZEDCHANGE 1702
Messages.WM_DWMWINDOWMAXIMIZEDCHANGE constant 1702
Messages.WM_ENABLE 1703
Messages.WM_ENABLE constant 1703
Messages.WM_ENDSESSION 1703
Messages.WM_ENDSESSION constant 1703
Messages.WM_ENTERIDLE 1703
Messages.WM_ENTERIDLE constant 1703
Messages.WM_ENTERMENULOOP 1704
Messages.WM_ENTERMENULOOP constant 1704
Messages.WM_ENTERSIZEMOVE 1704
Messages.WM_ENTERSIZEMOVE constant 1704
Messages.WM_ERASEBKGND 1705
Messages.WM_ERASEBKGND constant 1705
Messages.WM_EXITMENULOOP 1705
Messages.WM_EXITMENULOOP constant 1705
Messages.WM_EXITSIZEMOVE 1706
Messages.WM_EXITSIZEMOVE constant 1706
Messages.WM_FONTCHANGE 1706
Messages.WM_FONTCHANGE constant 1706
Messages.WM_GETDLGCODE 1707
Messages.WM_GETDLGCODE constant 1707
Messages.WM_GETFONT 1707
Messages.WM_GETFONT constant 1707
Messages.WM_GETHOTKEY 1707
Messages.WM_GETHOTKEY constant 1707
Messages.WM_GETICON 1708
Messages.WM_GETICON constant 1708
Messages.WM_GETMINMAXINFO 1708
Messages.WM_GETMINMAXINFO constant 1708
Messages.WM_GETOBJECT 1709
Messages.WM_GETOBJECT constant 1709
Messages.WM_GETTEXT 1709
Messages.WM_GETTEXT constant 1709
Messages.WM_GETTEXTLENGTH 1710
Messages.WM_GETTEXTLENGTH constant 1710
Messages.WM_HANDHELDFIRST 1710
Messages.WM_HANDHELDFIRST constant 1710
Messages.WM_HANDHELDLAST 1711
Messages.WM_HANDHELDLAST constant 1711
Messages.WM_HELP 1711
Messages.WM_HELP constant 1711
Messages.WM_HOTKEY 1711
Messages.WM_HOTKEY constant 1711
Messages.WM_HSCROLL 1712
Messages.WM_HSCROLL constant 1712
Messages.WM_HSCROLLCLIPBOARD 1712
Messages.WM_HSCROLLCLIPBOARD constant 1712
Messages.WM_ICONERASEBKGND 1713
Messages.WM_ICONERASEBKGND constant 1713
Messages.WM_IME_CHAR 1713
Messages.WM_IME_CHAR constant 1713
Messages.WM_IME_COMPOSITION 1714
Messages.WM_IME_COMPOSITION constant 1714
Messages.WM_IME_COMPOSITIONFULL 1714
Messages.WM_IME_COMPOSITIONFULL constant 1714
Messages.WM_IME_CONTROL 1715
Messages.WM_IME_CONTROL constant 1715
Messages.WM_IME_ENDCOMPOSITION 1715
Messages.WM_IME_ENDCOMPOSITION constant 1715
Messages.WM_IME_KEYDOWN 1715
Messages.WM_IME_KEYDOWN constant 1715
Messages.WM_IME_KEYLAST 1716
Messages.WM_IME_KEYLAST constant 1716
Messages.WM_IME_KEYUP 1716
Messages.WM_IME_KEYUP constant 1716
Messages.WM_IME_NOTIFY 1717
Messages.WM_IME_NOTIFY constant 1717
Messages.WM_IME_REQUEST 1717
Messages.WM_IME_REQUEST constant 1717

Messages.WM_IME_SELECT 1718
Messages.WM_IME_SELECT constant 1718
Messages.WM_IME_SETCONTEXT 1718
Messages.WM_IME_SETCONTEXT constant 1718
Messages.WM_IME_STARTCOMPOSITION 1718
Messages.WM_IME_STARTCOMPOSITION constant 1718
Messages.WM_INITDIALOG 1719
Messages.WM_INITDIALOG constant 1719
Messages.WM_INITMENU 1719
Messages.WM_INITMENU constant 1719
Messages.WM_INITMENUPOPUP 1720
Messages.WM_INITMENUPOPUP constant 1720
Messages.WM_INPUT 1720
Messages.WM_INPUT constant 1720
Messages.WM_INPUTLANGCHANGE 1721
Messages.WM_INPUTLANGCHANGE constant 1721
Messages.WM_INPUTLANGCHANGEREQUEST 1721
Messages.WM_INPUTLANGCHANGEREQUEST constant 1721
Messages.WM_KEYDOWN 1722
Messages.WM_KEYDOWN constant 1722
Messages.WM_KEYFIRST 1722
Messages.WM_KEYFIRST constant 1722
Messages.WM_KEYLAST 1722
Messages.WM_KEYLAST constant 1722
Messages.WM_KEYUP 1723
Messages.WM_KEYUP constant 1723
Messages.WM_KILLFOCUS 1723
Messages.WM_KILLFOCUS constant 1723
Messages.WM_LBUTTONDBLCLK 1724
Messages.WM_LBUTTONDBLCLK constant 1724
Messages.WM_LBUTTONDOWN 1724
Messages.WM_LBUTTONDOWN constant 1724
Messages.WM_LBUTTONUP 1725
Messages.WM_LBUTTONUP constant 1725
Messages.WM_MBUTTONDBLCLK 1725
Messages.WM_MBUTTONDBLCLK constant 1725
Messages.WM_MBUTTONDOWN 1726
Messages.WM_MBUTTONDOWN constant 1726
Messages.WM_MBUTTONUP 1726
Messages.WM_MBUTTONUP constant 1726
Messages.WM_MDIACTIVATE 1726
Messages.WM_MDIACTIVATE constant 1726
Messages.WM_MDICASCADE 1727
Messages.WM_MDICASCADE constant 1727
Messages.WM_MDICREATE 1727
Messages.WM_MDICREATE constant 1727
Messages.WM_MDIDESTROY 1728
Messages.WM_MDIDESTROY constant 1728
Messages.WM_MDIGETACTIVE 1728
Messages.WM_MDIGETACTIVE constant 1728
Messages.WM_MDIICONARRANGE 1729
Messages.WM_MDIICONARRANGE constant 1729
Messages.WM_MDIMAXIMIZE 1729
Messages.WM_MDIMAXIMIZE constant 1729
Messages.WM_MDINEXT 1729
Messages.WM_MDINEXT constant 1729
Messages.WM_MDIREFRESHMENU 1730
Messages.WM_MDIREFRESHMENU constant 1730
Messages.WM_MDIRESTORE 1730
Messages.WM_MDIRESTORE constant 1730
Messages.WM_MDISETMENU 1731
Messages.WM_MDISETMENU constant 1731
Messages.WM_MDITILE 1731
Messages.WM_MDITILE constant 1731
Messages.WM_MEASUREITEM 1732
Messages.WM_MEASUREITEM constant 1732
Messages.WM_MENUCHAR 1732
Messages.WM_MENUCHAR constant 1732
Messages.WM_MENUCOMMAND 1732
Messages.WM_MENUCOMMAND constant 1732
Messages.WM_MENUDRAG 1733
Messages.WM_MENUDRAG constant 1733
Messages.WM_MENUGETOBJECT 1733
Messages.WM_MENUGETOBJECT constant 1733
Messages.WM_MENURBUTTONUP 1734
Messages.WM_MENURBUTTONUP constant 1734
Messages.WM_MENUSELECT 1734
Messages.WM_MENUSELECT constant 1734

Messages.WM_MOUSEACTIVATE 1735
Messages.WM_MOUSEACTIVATE constant 1735
Messages.WM_MOUSEFIRST 1735
Messages.WM_MOUSEFIRST constant 1735
Messages.WM_MOUSEHOVER 1736
Messages.WM_MOUSEHOVER constant 1736
Messages.WM_MOUSELAST 1736
Messages.WM_MOUSELAST constant 1736
Messages.WM_MOUSELEAVE 1736
Messages.WM_MOUSELEAVE constant 1736
Messages.WM_MOUSEMOVE 1737
Messages.WM_MOUSEMOVE constant 1737
Messages.WM_MOUSEWHEEL 1737
Messages.WM_MOUSEWHEEL constant 1737
Messages.WM_MOVE 1738
Messages.WM_MOVE constant 1738
Messages.WM_MOVING 1738
Messages.WM_MOVING constant 1738
Messages.WM_NCACTIVATE 1739
Messages.WM_NCACTIVATE constant 1739
Messages.WM_NCCALCSIZE 1739
Messages.WM_NCCALCSIZE constant 1739
Messages.WM_NCCREATE 1739
Messages.WM_NCCREATE constant 1739
Messages.WM_NCDESTROY 1740
Messages.WM_NCDESTROY constant 1740
Messages.WM_NCHITTEST 1740
Messages.WM_NCHITTEST constant 1740
Messages.WM_NCLBUTTONDBLCLK 1741
Messages.WM_NCLBUTTONDBLCLK constant 1741
Messages.WM_NCLBUTTONDOWN 1741
Messages.WM_NCLBUTTONDOWN constant 1741
Messages.WM_NCLBUTTONUP 1742
Messages.WM_NCLBUTTONUP constant 1742
Messages.WM_NCMBUTTONDBLCLK 1742
Messages.WM_NCMBUTTONDBLCLK constant 1742
Messages.WM_NCMBUTTONDOWN 1742
Messages.WM_NCMBUTTONDOWN constant 1742
Messages.WM_NCMBUTTONUP 1743
Messages.WM_NCMBUTTONUP constant 1743
Messages.WM_NCMOUSEHOVER 1743
Messages.WM_NCMOUSEHOVER constant 1743
Messages.WM_NCMOUSELEAVE 1744
Messages.WM_NCMOUSELEAVE constant 1744
Messages.WM_NCMOUSEMOVE 1744
Messages.WM_NCMOUSEMOVE constant 1744
Messages.WM_NCPAINT 1745
Messages.WM_NCPAINT constant 1745
Messages.WM_NCRBUTTONDBLCLK 1745
Messages.WM_NCRBUTTONDBLCLK constant 1745
Messages.WM_NCRBUTTONDOWN 1746
Messages.WM_NCRBUTTONDOWN constant 1746
Messages.WM_NCRBUTTONUP 1746
Messages.WM_NCRBUTTONUP constant 1746
Messages.WM_NCXBUTTONDBLCLK 1746
Messages.WM_NCXBUTTONDBLCLK constant 1746
Messages.WM_NCXBUTTONDOWN 1747
Messages.WM_NCXBUTTONDOWN constant 1747
Messages.WM_NCXBUTTONUP 1747
Messages.WM_NCXBUTTONUP constant 1747
Messages.WM_NEXTDLGCTL 1748
Messages.WM_NEXTDLGCTL constant 1748
Messages.WM_NEXTMENU 1748
Messages.WM_NEXTMENU constant 1748
Messages.WM_NOTIFY 1749
Messages.WM_NOTIFY constant 1749
Messages.WM_NOTIFYFORMAT 1749
Messages.WM_NOTIFYFORMAT constant 1749
Messages.WM_NULL 1749
Messages.WM_NULL constant 1749
Messages.WM_PAINT 1750
Messages.WM_PAINT constant 1750
Messages.WM_PAINTCLIPBOARD 1750
Messages.WM_PAINTCLIPBOARD constant 1750
Messages.WM_PAINTICON 1751
Messages.WM_PAINTICON constant 1751
Messages.WM_PALETTECHANGED 1751
Messages.WM_PALETTECHANGED constant 1751

Messages.WM_PALETTEISCHANGING 1752
Messages.WM_PALETTEISCHANGING constant 1752
Messages.WM_PARENTNOTIFY 1752
Messages.WM_PARENTNOTIFY constant 1752
Messages.WM_PASTE 1752
Messages.WM_PASTE constant 1752
Messages.WM_PENWINFIRST 1753
Messages.WM_PENWINFIRST constant 1753
Messages.WM_PENWINLAST 1753
Messages.WM_PENWINLAST constant 1753
Messages.WM_POWER 1754
Messages.WM_POWER constant 1754
Messages.WM_POWERBROADCAST 1754
Messages.WM_POWERBROADCAST constant 1754
Messages.WM_PRINT 1755
Messages.WM_PRINT constant 1755
Messages.WM_PRINTCLIENT 1755
Messages.WM_PRINTCLIENT constant 1755
Messages.WM_QUERYDRAGICON 1756
Messages.WM_QUERYDRAGICON constant 1756
Messages.WM_QUERYENDSESSION 1756
Messages.WM_QUERYENDSESSION constant 1756
Messages.WM_QUERYNEWPALETTE 1756
Messages.WM_QUERYNEWPALETTE constant 1756
Messages.WM_QUERYOPEN 1757
Messages.WM_QUERYOPEN constant 1757
Messages.WM_QUERYUISTATE 1757
Messages.WM_QUERYUISTATE constant 1757
Messages.WM_QUEUESYNC 1758
Messages.WM_QUEUESYNC constant 1758
Messages.WM_QUIT 1758
Messages.WM_QUIT constant 1758
Messages.WM_RBUTTONDBLCLK 1759
Messages.WM_RBUTTONDBLCLK constant 1759
Messages.WM_RBUTTONDOWN 1759
Messages.WM_RBUTTONDOWN constant 1759
Messages.WM_RBUTTONUP 1759
Messages.WM_RBUTTONUP constant 1759
Messages.WM_RENDERALLFORMATS 1760
Messages.WM_RENDERALLFORMATS constant 1760
Messages.WM_RENDERFORMAT 1760
Messages.WM_RENDERFORMAT constant 1760
Messages.WM_SETCURSOR 1761
Messages.WM_SETCURSOR constant 1761
Messages.WM_SETFOCUS 1761
Messages.WM_SETFOCUS constant 1761
Messages.WM_SETFONT 1762
Messages.WM_SETFONT constant 1762
Messages.WM_SETHOTKEY 1762
Messages.WM_SETHOTKEY constant 1762
Messages.WM_SETICON 1763
Messages.WM_SETICON constant 1763
Messages.WM_SETREDRAW 1763
Messages.WM_SETREDRAW constant 1763
Messages.WM_SETTEXT 1763
Messages.WM_SETTEXT constant 1763
Messages.WM_SETTINGCHANGE 1764
Messages.WM_SETTINGCHANGE constant 1764
Messages.WM_SHOWWINDOW 1764
Messages.WM_SHOWWINDOW constant 1764
Messages.WM_SIZE 1765
Messages.WM_SIZE constant 1765
Messages.WM_SIZECLIPBOARD 1765
Messages.WM_SIZECLIPBOARD constant 1765
Messages.WM_SIZING 1766
Messages.WM_SIZING constant 1766
Messages.WM_SPOOLERSTATUS 1766
Messages.WM_SPOOLERSTATUS constant 1766
Messages.WM_STYLECHANGED 1766
Messages.WM_STYLECHANGED constant 1766
Messages.WM_STYLECHANGING 1767
Messages.WM_STYLECHANGING constant 1767
Messages.WM_SYSCHAR 1767
Messages.WM_SYSCHAR constant 1767
Messages.WM_SYSCOLORCHANGE 1768
Messages.WM_SYSCOLORCHANGE constant 1768
Messages.WM_SYSCOMMAND 1768
Messages.WM_SYSCOMMAND constant 1768

Messages.WM_SYSDEADCHAR 1769
Messages.WM_SYSDEADCHAR constant 1769
Messages.WM_SYSKEYDOWN 1769
Messages.WM_SYSKEYDOWN constant 1769
Messages.WM_SYSKEYUP 1770
Messages.WM_SYSKEYUP constant 1770
Messages.WM_SYSTEMERROR 1770
Messages.WM_SYSTEMERROR constant 1770
Messages.WM_TABLET_FIRST 1770
Messages.WM_TABLET_FIRST constant 1770
Messages.WM_TABLET_LAST 1771
Messages.WM_TABLET_LAST constant 1771
Messages.WM_TCARD 1771
Messages.WM_TCARD constant 1771
Messages.WM_THEMECHANGED 1772
Messages.WM_THEMECHANGED constant 1772
Messages.WM_TIMECHANGE 1772
Messages.WM_TIMECHANGE constant 1772
Messages.WM_TIMER 1773
Messages.WM_TIMER constant 1773
Messages.WM_UNDO 1773
Messages.WM_UNDO constant 1773
Messages.WM_UNICHAR 1774
Messages.WM_UNICHAR constant 1774
Messages.WM_UNINITMENUPOPUP 1774
Messages.WM_UNINITMENUPOPUP constant 1774
Messages.WM_UPDATEUISTATE 1774
Messages.WM_UPDATEUISTATE constant 1774
Messages.WM_USER 1775
Messages.WM_USER constant 1775
Messages.WM_USERCHANGED 1775
Messages.WM_USERCHANGED constant 1775
Messages.WM_VKEYTOITEM 1776
Messages.WM_VKEYTOITEM constant 1776
Messages.WM_VSCROLL 1776
Messages.WM_VSCROLL constant 1776
Messages.WM_VSCROLLCLIPBOARD 1777
Messages.WM_VSCROLLCLIPBOARD constant 1777
Messages.WM_WINDOWPOSCHANGED 1777
Messages.WM_WINDOWPOSCHANGED constant 1777
Messages.WM_WINDOWPOSCHANGING 1778
Messages.WM_WINDOWPOSCHANGING constant 1778
Messages.WM_WININICHANGE 1778
Messages.WM_WININICHANGE constant 1778
Messages.WM_WTSSESSION_CHANGE 1778
Messages.WM_WTSSESSION_CHANGE constant 1778
MetadataOnly enumeration member 477
Method
    TCustomQueryForm class 1846
    TQueryForm class 1975
MethodString
    TCustomQueryForm class 1846
mgColored enumeration member 2380
mgDisabled enumeration member 2380
mgEnabled enumeration member 2380
MidasCon 1781
MidasCon namespace 1781
    Classes 1782
    Structs, Records, Enums 1789
MidasCon.TConnectType 1789
MidasCon.TConnectType enumeration 1789
MidasCon.TMidasConnection 1782
MidasCon.TMidasConnection.ConnectType 1786
MidasCon.TMidasConnection.Create 1783
MidasCon.TMidasConnection.DoConnect 1784
MidasCon.TMidasConnection.DoDisconnect 1785
MidasCon.TMidasConnection.ServerPort 1786
MidasCon.TMidasConnection.SetConnectType 1785
MidasCon.TMidasConnection.SetUseBroker 1785
MidasCon.TMidasConnection.UseBroker 1787
MidasCon.TRemoteServer 1787
MiddleName
    TIBSecurityService class 449
    TUserInfo class 474
MidItems 1791
MidItems namespace 1791
    Classes 1796
    Constants 2242

Functions 2232
Interfaces 2200
Structs, Records, Enums 2237
Types 2240
MidItems.AddBoolAttrib 2232
MidItems.AddBoolAttrib function 2232
MidItems.AddCustomAttrib 2232
MidItems.AddCustomAttrib function 2232
MidItems.AddIntAttrib 2233
MidItems.AddIntAttrib function 2233
MidItems.AddQuotedAttrib 2233
MidItems.AddQuotedAttrib function 2233
MidItems.AddStringAttrib 2233
MidItems.AddStringAttrib function 2233
MidItems.CanAddClassHelper 2234
MidItems.CanAddClassHelper function 2234
MidItems.CreateDefaultButtonClasses 2234
MidItems.CreateDefaultButtonClasses function 2234
MidItems.DeclareSubmitForm 2234
MidItems.DeclareSubmitForm function 2234
MidItems.FindDispatcher 2235
MidItems.FindDispatcher function 2235
MidItems.FindProducer 2235
MidItems.FindProducer function 2235
MidItems.FormatColumnData 2235
MidItems.FormatColumnData function 2235
MidItems.FormatColumnHeading 2236
MidItems.FormatColumnHeading function 2236
MidItems.FormMethodNames 2242
MidItems.FormMethodNames constant 2242
MidItems.GetItemValuesFromDataSet 2236
MidItems.GetItemValuesFromDataSet function 2236
MidItems.HTMLAlignValues 2243
MidItems.HTMLAlignValues constant 2243
MidItems.HTMLVAlignValues 2243
MidItems.HTMLVAlignValues constant 2243
MidItems.IDataSetComponent 2200
MidItems.IDataSetComponent.DataSet 2202
MidItems.IDataSetComponent.GetDataSet 2201
MidItems.IDataSetComponent.GetXMLBroker 2201
MidItems.IDataSetComponent.GetXMLDataSet 2202
MidItems.IDataSetComponent.XMLBroker 2203
MidItems.IDataSetComponent.XMLDataSet 2203
MidItems.IDataSetField 2204
MidItems.IDataSetField.FieldName 2205
MidItems.IDataSetField.GetFieldName 2204
MidItems.IDataSetField.GetRowSetFieldAttributes 2205
MidItems.IDataSetField.SetFieldName 2205
MidItems.IDataSetFields 2206
MidItems.IDataSetFields.GetFieldsList 2207
MidItems.IDataSetFields.GetVisibleFields 2207
MidItems.IDataSetFields.HasStatusField 2207
MidItems.IDataSetFields.IsFieldInUse 2207
MidItems.IDataSetFields.ValidateDataSet 2207
MidItems.IDataSetFields.VisibleFields 2208
MidItems.IFormatColumn 2208
MidItems.IFormatColumn.FormatColumnData 2209
MidItems.IFormatColumn.FormatColumnHeading 2209
MidItems.IHTMLField 2209
MidItems.IHTMLField.GetHTMLControlName 2210
MidItems.IHTMLField.HTMLControlName 2211
MidItems.IHTMLForm 2211
MidItems.IHTMLForm.GetHTMLFormName 2212
MidItems.IHTMLForm.GetHTMLFormVarName 2212
MidItems.IHTMLForm.HTMLFormName 2213
MidItems.IHTMLForm.HTMLFormVarName 2213
MidItems.IInetXWebDataSet 2213
MidItems.IInetXWebDataSet.AddOnDataChange 2215
MidItems.IInetXWebDataSet.FetchParams 2215
MidItems.IInetXWebDataSet.FieldCount 2215
MidItems.IInetXWebDataSet.Fields 2216
MidItems.IInetXWebDataSet.ParamCount 2216
MidItems.IInetXWebDataSet.Params 2216
MidItems.IInetXWebDataSet.RemoveOnDataChange 2217
MidItems.IMidasWebDataSet 2240
MidItems.IMidasWebDataSet type 2240
MidItems.IQueryField 2217
MidItems.IQueryField.GetParamName 2218

MidItems.IQueryField.GetText 2218
MidItems.IQueryField.ParamName 2219
MidItems.IQueryField.SetParamName 2218
MidItems.IQueryField.SetText 2219
MidItems.IQueryField.Text 2219
MidItems.IQueryFields 2220
MidItems.IQueryFields.GetParamsList 2220
MidItems.IQueryFields.IsParamInUse 2221
MidItems.IQueryFields.ValidateDataSet 2221
MidItems.IQueryForm 2221
MidItems.IRestoreDefaults 2221
MidItems.IRestoreDefaults.RestoreDefaults 2222
MidItems.IStatusField 2223
MidItems.IValidateField 2223
MidItems.IValidateField.ValidateField 2224
MidItems.IValidateFields 2224
MidItems.IValidateFields.EnableValidateFields 2226
MidItems.IValidateFields.GetEnableValidateFields 2225
MidItems.IValidateFields.SetEnableValidateFields 2225
MidItems.IValidateFields.ValidateFields 2225
MidItems.IWebDisplay 2226
MidItems.IWebDisplay.GetIsMultipleRecordView 2227
MidItems.IWebDisplay.IsMultipleRecordView 2227
MidItems.IWebDisplayReference 2227
MidItems.IWebDisplayReference.GetWebDisplayComponent 2228
MidItems.IWebDisplayReference.WebDisplayComponent 2228
MidItems.IXMLDisplay 2229
MidItems.IXMLDisplay.GetIsMultipleRecordView 2229
MidItems.IXMLDisplay.GetXMLDisplayName 2230
MidItems.IXMLDisplay.GetXMLRowSetName 2230
MidItems.IXMLDisplay.IsMultipleRecordView 2230
MidItems.IXMLDisplay.XMLDisplayName 2231
MidItems.IXMLDisplay.XMLRowSetName 2231
MidItems.sActionAttr 2243
MidItems.sActionAttr constant 2243
MidItems.sAlignAttr 2244
MidItems.sAlignAttr constant 2244
MidItems.sBgColorAttr 2244
MidItems.sBgColorAttr constant 2244
MidItems.sBorderAttr 2244
MidItems.sBorderAttr constant 2244
MidItems.sCellPaddingAttr 2245
MidItems.sCellPaddingAttr constant 2245
MidItems.sCellSpacingAttr 2245
MidItems.sCellSpacingAttr constant 2245
MidItems.sCheckedAttr 2245
MidItems.sCheckedAttr constant 2245
MidItems.sClassAttr 2246
MidItems.sClassAttr constant 2246
MidItems.sColsAttr 2246
MidItems.sColsAttr constant 2246
MidItems.sColSpanAttr 2246
MidItems.sColSpanAttr constant 2246
MidItems.sEncTypeAttr 2247
MidItems.sEncTypeAttr constant 2247
MidItems.sImageHeightAttr 2247
MidItems.sImageHeightAttr constant 2247
MidItems.sImageWidthAttr 2247
MidItems.sImageWidthAttr constant 2247
MidItems.sMaxLengthAttr 2248
MidItems.sMaxLengthAttr constant 2248
MidItems.sMethodAttr 2248
MidItems.sMethodAttr constant 2248
MidItems.sMultipleAttr 2248
MidItems.sMultipleAttr constant 2248
MidItems.sNameAttr 2249
MidItems.sNameAttr constant 2249
MidItems.sOnClickAttr 2249
MidItems.sOnClickAttr constant 2249
MidItems.sReadOnlyAttr 2249
MidItems.sReadOnlyAttr constant 2249
MidItems.sRowsAttr 2250
MidItems.sRowsAttr constant 2250
MidItems.sSelectedAttr 2250
MidItems.sSelectedAttr constant 2250
MidItems.sSizeAttr 2250
MidItems.sSizeAttr constant 2250
MidItems.sStyleAttr 2251

MidItems.sStyleAttr constant 2251
MidItems.sTabIndexAttr 2251
MidItems.sTabIndexAttr constant 2251
MidItems.sTypeAttr 2251
MidItems.sTypeAttr constant 2251
MidItems.sVAlignAttr 2252
MidItems.sVAlignAttr constant 2252
MidItems.sValueAttr 2252
MidItems.sValueAttr constant 2252
MidItems.sWidthAttr 2252
MidItems.sWidthAttr constant 2252
MidItems.sWrapAttr 2253
MidItems.sWrapAttr constant 2253
MidItems.TApplyUpdatesButton 1798
MidItems.TApplyUpdatesButton.AddElements 1800
MidItems.TApplyUpdatesButton.Caption 1801
MidItems.TApplyUpdatesButton.Create 1799
MidItems.TApplyUpdatesButton.Custom 1802
MidItems.TApplyUpdatesButton.GetSubComponents 1800
MidItems.TApplyUpdatesButton.ImplAddElements 1800
MidItems.TApplyUpdatesButton.ImplContent 1800
MidItems.TApplyUpdatesButton.Style 1802
MidItems.TApplyUpdatesButton.StyleRule 1802
MidItems.TApplyUpdatesButton.XMLBroker 1803
MidItems.TApplyUpdatesButton.XMLUseParent 1803
MidItems.TCaptionAttributes 1804
MidItems.TCaptionAttributes.Align 1805
MidItems.TCaptionAttributes.Create 1805
MidItems.TCaptionAttributes.Custom 1806
MidItems.TCaptionAttributes.Parent 1806
MidItems.TCaptionAttributes.Style 1806
MidItems.TCaptionAttributes.StyleRule 1807
MidItems.TCaptionAttributes.VAlign 1807
MidItems.TCaptionPosition 2237
MidItems.TCaptionPosition enumeration 2237
MidItems.TCustomDataForm 1808
MidItems.TCustomDataForm.GetHTMLFormTag 1810
MidItems.TCustomDataForm.ImplCanAddClass 1810
MidItems.TCustomDataForm.ImplContent 1810
MidItems.TCustomDataGrid 1811
MidItems.TCustomDataGrid.Create 1814
MidItems.TCustomDataGrid.Destroy 1814
MidItems.TCustomDataGrid.DisplayRows 1817
MidItems.TCustomDataGrid.FormatTable 1814
MidItems.TCustomDataGrid.HeadingAttributes 1817
MidItems.TCustomDataGrid.ImplAddElements 1815
MidItems.TCustomDataGrid.ImplCanAddClass 1815
MidItems.TCustomDataGrid.ImplContent 1815
MidItems.TCustomDataGrid.ImplIsMultipleRecordView 1816
MidItems.TCustomDataGrid.RowAttributes 1818
MidItems.TCustomDataGrid.SetHeadingAttributes 1816
MidItems.TCustomDataGrid.SetRowAttributes 1816
MidItems.TCustomDataGrid.SetTableAttributes 1817
MidItems.TCustomDataGrid.TableAttributes 1818
MidItems.TCustomDataNavigator 1819
MidItems.TCustomDataNavigator.Custom 1823
MidItems.TCustomDataNavigator.GetDefaultButtons 1821
MidItems.TCustomDataNavigator.ImplCanAddClass 1821
MidItems.TCustomDataNavigator.ImplContent 1821
MidItems.TCustomDataNavigator.ImplGetSubComponents 1822
MidItems.TCustomDataNavigator.Style 1823
MidItems.TCustomDataNavigator.StyleRule 1824
MidItems.TCustomDataNavigator.VisibleButtons 1822
MidItems.TCustomFieldGroup 1824
MidItems.TCustomFieldGroup.Create 1827
MidItems.TCustomFieldGroup.Custom 1829
MidItems.TCustomFieldGroup.Destroy 1827
MidItems.TCustomFieldGroup.GetLayoutAttributes 1828
MidItems.TCustomFieldGroup.ImplContent 1828
MidItems.TCustomFieldGroup.Style 1829
MidItems.TCustomFieldGroup.StyleRule 1830
MidItems.TCustomLayoutGroup 1830
MidItems.TCustomLayoutGroup.Create 1832
MidItems.TCustomLayoutGroup.Custom 1834
MidItems.TCustomLayoutGroup.Destroy 1832
MidItems.TCustomLayoutGroup.DisplayColumns 1834
MidItems.TCustomLayoutGroup.GetLayoutAttributes 1832
MidItems.TCustomLayoutGroup.ImplCanAddClass 1833

MidItems.TCustomLayoutGroup.ImplContent 1833
MidItems.TCustomLayoutGroup.Style 1835
MidItems.TCustomLayoutGroup.StyleRule 1835
MidItems.TCustomQueryButtons 1836
MidItems.TCustomQueryButtons.Custom 1839
MidItems.TCustomQueryButtons.GetDefaultButtons 1837
MidItems.TCustomQueryButtons.ImplAddElements 1838
MidItems.TCustomQueryButtons.ImplCanAddClass 1838
MidItems.TCustomQueryButtons.ImplContent 1838
MidItems.TCustomQueryButtons.Style 1840
MidItems.TCustomQueryButtons.StyleRule 1840
MidItems.TCustomQueryButtons.VisibleButtons 1839
MidItems.TCustomQueryForm 1841
MidItems.TCustomQueryForm.Action 1846
MidItems.TCustomQueryForm.AssignStrings 1843
MidItems.TCustomQueryForm.AssignStringsCallback 1844
MidItems.TCustomQueryForm.Create 1843
MidItems.TCustomQueryForm.GetHTMLFormTag 1844
MidItems.TCustomQueryForm.ImplAddElements 1844
MidItems.TCustomQueryForm.ImplCanAddClass 1845
MidItems.TCustomQueryForm.ImplContent 1845
MidItems.TCustomQueryForm.Method 1846
MidItems.TCustomQueryForm.MethodString 1846
MidItems.TDataForm 1847
MidItems.TDataForm.Custom 1849
MidItems.TDataForm.Style 1849
MidItems.TDataForm.StyleRule 1850
MidItems.TDataGrid 1850
MidItems.TDataGrid.DisplayRows 1853
MidItems.TDataGrid.HeadingAttributes 1854
MidItems.TDataGrid.RowAttributes 1854
MidItems.TDataGrid.TableAttributes 1855
MidItems.TDataNavigator 1855
MidItems.TDataNavigator.Custom 1857
MidItems.TDataNavigator.Style 1858
MidItems.TDataNavigator.StyleRule 1858
MidItems.TDataNavigator.XMLComponent 1859
MidItems.TDataSetButton 1859
MidItems.TDataSetButton.AddElements 1862
MidItems.TDataSetButton.Caption 1864
MidItems.TDataSetButton.Custom 1865
MidItems.TDataSetButton.DefaultCaption 1862
MidItems.TDataSetButton.GetCaption 1862
MidItems.TDataSetButton.GetMethodName 1863
MidItems.TDataSetButton.GetSubComponents 1863
MidItems.TDataSetButton.ImplAddElements 1863
MidItems.TDataSetButton.ImplContent 1863
MidItems.TDataSetButton.Style 1865
MidItems.TDataSetButton.StyleRule 1865
MidItems.TDataSetButton.XMLComponent 1866
MidItems.TDataSetButton.XMLMethodName 1862
MidItems.TDataSetButton.XMLUseParent 1866
MidItems.TDeleteButton 1867
MidItems.TDeleteButton.Create 1869
MidItems.TextAreaWrap 2253
MidItems.TextAreaWrap constant 2253
MidItems.TFieldGroup 1869
MidItems.TFieldGroup.Custom 1873
MidItems.TFieldGroup.ImplAddElements 1872
MidItems.TFieldGroup.ImplCanAddClass 1872
MidItems.TFieldGroup.Style 1873
MidItems.TFieldGroup.StyleRule 1874
MidItems.TFieldRadioGroup 1874
MidItems.TFieldRadioGroup.Caption 1878
MidItems.TFieldRadioGroup.CaptionAttributes 1878
MidItems.TFieldRadioGroup.CaptionPosition 1878
MidItems.TFieldRadioGroup.Custom 1879
MidItems.TFieldRadioGroup.DataSet 1879
MidItems.TFieldRadioGroup.DisplayColumns 1880
MidItems.TFieldRadioGroup.DisplayWidth 1880
MidItems.TFieldRadioGroup.FieldName 1880
MidItems.TFieldRadioGroup.Items 1881
MidItems.TFieldRadioGroup.ItemsField 1881
MidItems.TFieldRadioGroup.ReadOnly 1882
MidItems.TFieldRadioGroup.Style 1882
MidItems.TFieldRadioGroup.StyleRule 1883
MidItems.TFieldRadioGroup.TabIndex 1883
MidItems.TFieldRadioGroup.Values 1883

MidItems.TFieldRadioGroup.ValuesField 1884
MidItems.TFieldSelectOptions 1884
MidItems.TFieldSelectOptions.Caption 1888
MidItems.TFieldSelectOptions.CaptionAttributes 1888
MidItems.TFieldSelectOptions.CaptionPosition 1888
MidItems.TFieldSelectOptions.Custom 1889
MidItems.TFieldSelectOptions.DataSet 1889
MidItems.TFieldSelectOptions.DisplayRows 1890
MidItems.TFieldSelectOptions.FieldName 1890
MidItems.TFieldSelectOptions.Items 1891
MidItems.TFieldSelectOptions.ItemsField 1891
MidItems.TFieldSelectOptions.Style 1891
MidItems.TFieldSelectOptions.StyleRule 1892
MidItems.TFieldSelectOptions.TabIndex 1892
MidItems.TFieldSelectOptions.Values 1893
MidItems.TFieldSelectOptions.ValuesField 1893
MidItems.TFieldStatus 1894
MidItems.TFieldStatus.Caption 1896
MidItems.TFieldStatus.CaptionAttributes 1896
MidItems.TFieldStatus.CaptionPosition 1897
MidItems.TFieldStatus.Custom 1897
MidItems.TFieldStatus.Style 1897
MidItems.TFieldStatus.StyleRule 1898
MidItems.TFieldText 1899
MidItems.TFieldText.Caption 1901
MidItems.TFieldText.CaptionAttributes 1902
MidItems.TFieldText.CaptionPosition 1902
MidItems.TFieldText.Custom 1902
MidItems.TFieldText.DisplayWidth 1903
MidItems.TFieldText.FieldName 1903
MidItems.TFieldText.ReadOnly 1904
MidItems.TFieldText.Style 1904
MidItems.TFieldText.StyleRule 1904
MidItems.TFieldText.TabIndex 1905
MidItems.TFieldTextArea 1905
MidItems.TFieldTextArea.Caption 1908
MidItems.TFieldTextArea.CaptionAttributes 1908
MidItems.TFieldTextArea.CaptionPosition 1909
MidItems.TFieldTextArea.Custom 1909
MidItems.TFieldTextArea.DisplayRows 1910
MidItems.TFieldTextArea.DisplayWidth 1910
MidItems.TFieldTextArea.FieldName 1910
MidItems.TFieldTextArea.ReadOnly 1911
MidItems.TFieldTextArea.Style 1911
MidItems.TFieldTextArea.StyleRule 1911
MidItems.TFieldTextArea.TabIndex 1912
MidItems.TFieldTextArea.Wrap 1912
MidItems.TFirstButton 1913
MidItems.TFirstButton.Create 1915
MidItems.TFormLayout 1915
MidItems.TFormLayout.BreakButtons 1922
MidItems.TFormLayout.ColumnCount 1922
MidItems.TFormLayout.Create 1916
MidItems.TFormLayout.EndButtons 1917
MidItems.TFormLayout.EndFields 1917
MidItems.TFormLayout.EndLayout 1917
MidItems.TFormLayout.EndRow 1918
MidItems.TFormLayout.EndTable 1918
MidItems.TFormLayout.ImplLayoutButton 1918
MidItems.TFormLayout.ImplLayoutField 1919
MidItems.TFormLayout.ImplLayoutLabelAndField 1919
MidItems.TFormLayout.ImplLayoutTable 1920
MidItems.TFormLayout.NextColumn 1920
MidItems.TFormLayout.StartButtons 1920
MidItems.TFormLayout.StartFields 1921
MidItems.TFormLayout.StartRow 1921
MidItems.TFormLayout.StartTable 1921
MidItems.TFormLayout.TableHeader 1922
MidItems.TFormMethod 2238
MidItems.TFormMethod enumeration 2238
MidItems.TGridAttributes 1923
MidItems.TGridAttributes.Align 1926
MidItems.TGridAttributes.AssignTo 1924
MidItems.TGridAttributes.BgColor 1926
MidItems.TGridAttributes.Border 1926
MidItems.TGridAttributes.CellPadding 1927
MidItems.TGridAttributes.CellSpacing 1927
MidItems.TGridAttributes.Create 1924

MidItems.TGridAttributes.Custom 1928
MidItems.TGridAttributes.Parent 1928
MidItems.TGridAttributes.Style 1928
MidItems.TGridAttributes.StyleRule 1929
MidItems.TGridRowAttributes 1929
MidItems.TGridRowAttributes.Align 1932
MidItems.TGridRowAttributes.AssignTo 1931
MidItems.TGridRowAttributes.BgColor 1932
MidItems.TGridRowAttributes.Create 1930
MidItems.TGridRowAttributes.Custom 1933
MidItems.TGridRowAttributes.Parent 1933
MidItems.TGridRowAttributes.Style 1934
MidItems.TGridRowAttributes.StyleRule 1934
MidItems.TGridRowAttributes.VAlign 1934
MidItems.TInsertButton 1935
MidItems.TInsertButton.Create 1937
MidItems.TLastButton 1937
MidItems.TLastButton.Create 1939
MidItems.TLayoutGroup 1939
MidItems.TLayoutGroup.Custom 1941
MidItems.TLayoutGroup.DisplayColumns 1941
MidItems.TLayoutGroup.Style 1942
MidItems.TLayoutGroup.StyleRule 1942
MidItems.TLayoutState 2238
MidItems.TLayoutState enumeration 2238
MidItems.TNextButton 1943
MidItems.TNextButton.Create 1944
MidItems.TNextPageButton 1945
MidItems.TNextPageButton.Create 1947
MidItems.TPostButton 1947
MidItems.TPostButton.Create 1949
MidItems.TPriorButton 1949
MidItems.TPriorButton.Create 1951
MidItems.TPriorPageButton 1951
MidItems.TPriorPageButton.Create 1953
MidItems.TQueryButton 1953
MidItems.TQueryButton.Caption 1959
MidItems.TQueryButton.Create 1955
MidItems.TQueryButton.Custom 1960
MidItems.TQueryButton.DefaultCaption 1955
MidItems.TQueryButton.EventContent 1956
MidItems.TQueryButton.GetCaption 1956
MidItems.TQueryButton.GetHTMLControlName 1957
MidItems.TQueryButton.GetHTMLForm 1957
MidItems.TQueryButton.GetInputType 1957
MidItems.TQueryButton.GetXMLComponent 1957
MidItems.TQueryButton.ImplContent 1958
MidItems.TQueryButton.InputType 1956
MidItems.TQueryButton.Notification 1958
MidItems.TQueryButton.ParentXMLComponent 1959
MidItems.TQueryButton.SetXMLComponent 1959
MidItems.TQueryButton.SetXMLUseParent 1959
MidItems.TQueryButton.Style 1960
MidItems.TQueryButton.StyleRule 1961
MidItems.TQueryButton.XMLComponent 1961
MidItems.TQueryButton.XMLUseParent 1962
MidItems.TQueryButtons 1962
MidItems.TQueryButtons.Custom 1964
MidItems.TQueryButtons.Style 1964
MidItems.TQueryButtons.StyleRule 1965
MidItems.TQueryFieldGroup 1965
MidItems.TQueryFieldGroup.Custom 1970
MidItems.TQueryFieldGroup.FindParam 1969
MidItems.TQueryFieldGroup.GetParamsList 1969
MidItems.TQueryFieldGroup.ImplCanAddClass 1969
MidItems.TQueryFieldGroup.IsParamInUse 1970
MidItems.TQueryFieldGroup.Style 1971
MidItems.TQueryFieldGroup.StyleRule 1971
MidItems.TQueryForm 1971
MidItems.TQueryForm.Action 1974
MidItems.TQueryForm.Custom 1974
MidItems.TQueryForm.Method 1975
MidItems.TQueryForm.Style 1975
MidItems.TQueryForm.StyleRule 1975
MidItems.TQueryRadioGroup 1976
MidItems.TQueryRadioGroup.AddAttributes 1979
MidItems.TQueryRadioGroup.Caption 1981
MidItems.TQueryRadioGroup.CaptionAttributes 1981

MidItems.TQueryRadioGroup.CaptionPosition 1982
MidItems.TQueryRadioGroup.Custom 1982
MidItems.TQueryRadioGroup.DataSet 1982
MidItems.TQueryRadioGroup.DisplayColumns 1983
MidItems.TQueryRadioGroup.DisplayWidth 1983
MidItems.TQueryRadioGroup.FieldName 1984
MidItems.TQueryRadioGroup.GetCheckIndex 1979
MidItems.TQueryRadioGroup.GetText 1980
MidItems.TQueryRadioGroup.IsQueryField 1980
MidItems.TQueryRadioGroup.Items 1984
MidItems.TQueryRadioGroup.ItemsField 1985
MidItems.TQueryRadioGroup.ParamName 1985
MidItems.TQueryRadioGroup.ReadOnly 1986
MidItems.TQueryRadioGroup.SetText 1980
MidItems.TQueryRadioGroup.Style 1986
MidItems.TQueryRadioGroup.StyleRule 1986
MidItems.TQueryRadioGroup.TabIndex 1987
MidItems.TQueryRadioGroup.Text 1987
MidItems.TQueryRadioGroup.Values 1988
MidItems.TQueryRadioGroup.ValuesField 1988
MidItems.TQuerySelectOptions 1989
MidItems.TQuerySelectOptions.Caption 1993
MidItems.TQuerySelectOptions.CaptionAttributes 1994
MidItems.TQuerySelectOptions.CaptionPosition 1994
MidItems.TQuerySelectOptions.Custom 1994
MidItems.TQuerySelectOptions.DataSet 1995
MidItems.TQuerySelectOptions.DisplayRows 1995
MidItems.TQuerySelectOptions.FieldName 1996
MidItems.TQuerySelectOptions.GetSelectIndex 1992
MidItems.TQuerySelectOptions.GetText 1992
MidItems.TQuerySelectOptions.IsQueryField 1992
MidItems.TQuerySelectOptions.Items 1996
MidItems.TQuerySelectOptions.ItemsField 1997
MidItems.TQuerySelectOptions.ParamName 1997
MidItems.TQuerySelectOptions.SetText 1993
MidItems.TQuerySelectOptions.Style 1998
MidItems.TQuerySelectOptions.StyleRule 1998
MidItems.TQuerySelectOptions.TabIndex 1999
MidItems.TQuerySelectOptions.Text 1999
MidItems.TQuerySelectOptions.Values 1999
MidItems.TQuerySelectOptions.ValuesField 2000
MidItems.TQueryText 2000
MidItems.TQueryText.AddAttributes 2004
MidItems.TQueryText.Caption 2005
MidItems.TQueryText.CaptionAttributes 2005
MidItems.TQueryText.CaptionPosition 2006
MidItems.TQueryText.Create 2003
MidItems.TQueryText.Custom 2006
MidItems.TQueryText.DisplayWidth 2007
MidItems.TQueryText.FieldName 2007
MidItems.TQueryText.GetText 2004
MidItems.TQueryText.IsQueryField 2004
MidItems.TQueryText.MaxWidth 2007
MidItems.TQueryText.ParamName 2008
MidItems.TQueryText.ReadOnly 2008
MidItems.TQueryText.SetText 2004
MidItems.TQueryText.Style 2009
MidItems.TQueryText.StyleRule 2009
MidItems.TQueryText.TabIndex 2010
MidItems.TQueryText.Text 2010
MidItems.TQueryTextArea 2010
MidItems.TQueryTextArea.AddAttributes 2014
MidItems.TQueryTextArea.Caption 2016
MidItems.TQueryTextArea.CaptionAttributes 2016
MidItems.TQueryTextArea.CaptionPosition 2017
MidItems.TQueryTextArea.ControlContent 2014
MidItems.TQueryTextArea.Create 2013
MidItems.TQueryTextArea.Custom 2017
MidItems.TQueryTextArea.Destroy 2013
MidItems.TQueryTextArea.DisplayRows 2017
MidItems.TQueryTextArea.DisplayWidth 2018
MidItems.TQueryTextArea.FieldName 2018
MidItems.TQueryTextArea.GetText 2014
MidItems.TQueryTextArea.IsQueryField 2015
MidItems.TQueryTextArea.Lines 2019
MidItems.TQueryTextArea.ParamName 2019
MidItems.TQueryTextArea.ReadOnly 2020
MidItems.TQueryTextArea.SetLines 2015

MidItems.TQueryTextArea.SetText 2015
MidItems.TQueryTextArea.Style 2020
MidItems.TQueryTextArea.StyleRule 2020
MidItems.TQueryTextArea.TabIndex 2021
MidItems.TQueryTextArea.Wrap 2021
MidItems.TResetQueryButton 2022
MidItems.TResetQueryButton.Create 2023
MidItems.TResetQueryButton.EventContent 2024
MidItems.TSelectOptionsColumn 2024
MidItems.TSelectOptionsColumn.Caption 2028
MidItems.TSelectOptionsColumn.CaptionAttributes 2028
MidItems.TSelectOptionsColumn.Custom 2028
MidItems.TSelectOptionsColumn.DataSet 2029
MidItems.TSelectOptionsColumn.DisplayRows 2029
MidItems.TSelectOptionsColumn.FieldName 2030
MidItems.TSelectOptionsColumn.IsColumn 2027
MidItems.TSelectOptionsColumn.Items 2030
MidItems.TSelectOptionsColumn.ItemsField 2031
MidItems.TSelectOptionsColumn.Style 2031
MidItems.TSelectOptionsColumn.StyleRule 2032
MidItems.TSelectOptionsColumn.TabIndex 2032
MidItems.TSelectOptionsColumn.Values 2032
MidItems.TSelectOptionsColumn.ValuesField 2033
MidItems.TStatusColumn 2033
MidItems.TStatusColumn.Caption 2036
MidItems.TStatusColumn.CaptionAttributes 2036
MidItems.TStatusColumn.Custom 2037
MidItems.TStatusColumn.IsColumn 2035
MidItems.TStatusColumn.Style 2037
MidItems.TStatusColumn.StyleRule 2038
MidItems.TSubmitQueryButton 2038
MidItems.TSubmitQueryButton.Create 2040
MidItems.TSubmitQueryButton.EventContent 2040
MidItems.TTextAreaColumn 2041
MidItems.TTextAreaColumn.Caption 2044
MidItems.TTextAreaColumn.CaptionAttributes 2044
MidItems.TTextAreaColumn.Custom 2045
MidItems.TTextAreaColumn.DisplayRows 2045
MidItems.TTextAreaColumn.DisplayWidth 2046
MidItems.TTextAreaColumn.FieldName 2046
MidItems.TTextAreaColumn.IsColumn 2043
MidItems.TTextAreaColumn.ReadOnly 2046
MidItems.TTextAreaColumn.Style 2047
MidItems.TTextAreaColumn.StyleRule 2047
MidItems.TTextAreaColumn.TabIndex 2048
MidItems.TTextAreaColumn.Wrap 2048
MidItems.TTextAreaWrap 2239
MidItems.TTextAreaWrap enumeration 2239
MidItems.TTextColumn 2049
MidItems.TTextColumn.Caption 2052
MidItems.TTextColumn.CaptionAttributes 2052
MidItems.TTextColumn.Custom 2053
MidItems.TTextColumn.DisplayWidth 2053
MidItems.TTextColumn.FieldName 2053
MidItems.TTextColumn.IsColumn 2051
MidItems.TTextColumn.ReadOnly 2054
MidItems.TTextColumn.Style 2054
MidItems.TTextColumn.StyleRule 2055
MidItems.TTextColumn.TabIndex 2055
MidItems.TUndoButton 2055
MidItems.TUndoButton.Create 2057
MidItems.TWebButton 2058
MidItems.TWebButton.Caption 2062
MidItems.TWebButton.Content 2060
MidItems.TWebButton.Create 2059
MidItems.TWebButton.Custom 2062
MidItems.TWebButton.GetCaption 2060
MidItems.TWebButton.GetLayoutAttributes 2060
MidItems.TWebButton.ImplContent 2061
MidItems.TWebButton.SetCaption 2061
MidItems.TWebButton.Style 2063
MidItems.TWebButton.StyleRule 2063
MidItems.TWebButtonClass 2240
MidItems.TWebButtonClass type 2240
MidItems.TWebControlGroup 2064
MidItems.TWebControlGroup.AddElements 2066
MidItems.TWebControlGroup.CanAddClass 2067
MidItems.TWebControlGroup.ChangeFieldControls 2067

MidItems.TWebControlGroup.Content 2068
MidItems.TWebControlGroup.Create 2065
MidItems.TWebControlGroup.DefaultWebComponents 2072
MidItems.TWebControlGroup.Destroy 2066
MidItems.TWebControlGroup.GetDefaultWebComponents 2068
MidItems.TWebControlGroup.GetLayoutAttributes 2069
MidItems.TWebControlGroup.GetSubComponents 2069
MidItems.TWebControlGroup.ImplAddElements 2069
MidItems.TWebControlGroup.ImplCanAddClass 2070
MidItems.TWebControlGroup.ImplContent 2070
MidItems.TWebControlGroup.ImplGetSubComponents 2071
MidItems.TWebControlGroup.SetDefaultWebComponents 2071
MidItems.TWebControlGroup.WebFieldControls 2072
MidItems.TWebDataDisplay 2073
MidItems.TWebDataDisplay.Caption 2083
MidItems.TWebDataDisplay.CaptionAttributes 2084
MidItems.TWebDataDisplay.CaptionPosition 2084
MidItems.TWebDataDisplay.Content 2076
MidItems.TWebDataDisplay.ControlContent 2077
MidItems.TWebDataDisplay.Create 2075
MidItems.TWebDataDisplay.Custom 2084
MidItems.TWebDataDisplay.Destroy 2075
MidItems.TWebDataDisplay.EventContent 2077
MidItems.TWebDataDisplay.FormatCaption 2077
MidItems.TWebDataDisplay.GetCaption 2078
MidItems.TWebDataDisplay.GetHTMLControlName 2078
MidItems.TWebDataDisplay.GetHTMLForm 2078
MidItems.TWebDataDisplay.GetLayoutAttributes 2079
MidItems.TWebDataDisplay.GetXmlDisplayName 2079
MidItems.TWebDataDisplay.GetXmlRowSetName 2080
MidItems.TWebDataDisplay.ImplContent 2080
MidItems.TWebDataDisplay.ImplGetHTMLControlName 2081
MidItems.TWebDataDisplay.IsColumn 2081
MidItems.TWebDataDisplay.IsQueryField 2081
MidItems.TWebDataDisplay.LabelContent 2082
MidItems.TWebDataDisplay.LayoutAttributes 2085
MidItems.TWebDataDisplay.SetCaption 2082
MidItems.TWebDataDisplay.SetCaptionAttributes 2083
MidItems.TWebDataDisplay.Style 2085
MidItems.TWebDataDisplay.StyleRule 2086
MidItems.TWebDataDisplay.TabIndex 2086
MidItems.TWebDataDisplayClass 2241
MidItems.TWebDataDisplayClass type 2241
MidItems.TWebDataInput 2087
MidItems.TWebDataInput.EventContent 2089
MidItems.TWebDataInput.FieldName 2094
MidItems.TWebDataInput.FindAssociatedField 2090
MidItems.TWebDataInput.GetCaption 2090
MidItems.TWebDataInput.GetFieldName 2090
MidItems.TWebDataInput.GetParamName 2090
MidItems.TWebDataInput.GetRowSetFieldAttributes 2091
MidItems.TWebDataInput.ImplGetFieldName 2091
MidItems.TWebDataInput.ImplGetHTMLControlName 2091
MidItems.TWebDataInput.ImplGetRowSetFieldAttributes 2091
MidItems.TWebDataInput.ImplRestoreDefaults 2092
MidItems.TWebDataInput.ParamName 2094
MidItems.TWebDataInput.RestoreDefaults 2092
MidItems.TWebDataInput.RestoreFieldDefaults 2092
MidItems.TWebDataInput.SetCaption 2092
MidItems.TWebDataInput.SetFieldName 2093
MidItems.TWebDataInput.SetParamName 2093
MidItems.TWebDataInput.ValidateField 2093
MidItems.TWebDataInputClass 2241
MidItems.TWebDataInputClass type 2241
MidItems.TWebDisplay 2095
MidItems.TWebDisplay.Changed 2096
MidItems.TWebDisplay.ChangedDisplayComponent 2096
MidItems.TWebDisplay.Create 2095
MidItems.TWebDisplay.DisplayComponent 2097
MidItems.TWebDisplay.GetDisplayComponent 2096
MidItems.TWebDisplay.Notification 2097
MidItems.TWebDisplay.OnChange 2098
MidItems.TWebDisplay.Parent 2097
MidItems.TWebDisplay.SetDisplayComponent 2097
MidItems.TWebDisplayReferenceGroup 2098
MidItems.TWebDisplayReferenceGroup.Create 2100
MidItems.TWebDisplayReferenceGroup.Destroy 2100
MidItems.TWebDisplayReferenceGroup.GetWebDisplayComponent

2100
MidItems.TWebDisplayReferenceGroup.Notification 2101
MidItems.TWebDisplayReferenceGroup.SetWebDisplayComponent 2101
MidItems.TWebDisplayReferenceGroup.WebDisplay 2102
MidItems.TWebDisplayReferenceGroup.WebDisplayComponent 2102
MidItems.TWebForm 2102
MidItems.TWebForm.AddElements 2105
MidItems.TWebForm.CanAddClass 2105
MidItems.TWebForm.Content 2106
MidItems.TWebForm.Create 2104
MidItems.TWebForm.Custom 2110
MidItems.TWebForm.Destroy 2104
MidItems.TWebForm.GetDefaultWebComponents 2106
MidItems.TWebForm.GetHTMLFormName 2107
MidItems.TWebForm.GetHTMLFormVarName 2107
MidItems.TWebForm.GetLayoutAttributes 2107
MidItems.TWebForm.GetSubComponents 2108
MidItems.TWebForm.GetVisibleFields 2108
MidItems.TWebForm.ImplAddElements 2108
MidItems.TWebForm.ImplCanAddClass 2109
MidItems.TWebForm.ImplContent 2109
MidItems.TWebForm.SetDefaultWebComponents 2110
MidItems.TWebForm.Style 2111
MidItems.TWebForm.StyleRule 2111
MidItems.TWebForm.VisibleFields 2112
MidItems.TWebForm.WebComponents 2112
MidItems.TWebListInput 2113
MidItems.TWebListInput.AddElements 2117
MidItems.TWebListInput.ControlContent 2117
MidItems.TWebListInput.Create 2116
MidItems.TWebListInput.DataSet 2119
MidItems.TWebListInput.Destroy 2116
MidItems.TWebListInput.FormatInputs 2117
MidItems.TWebListInput.GetSubComponents 2118
MidItems.TWebListInput.ImplAddElements 2118
MidItems.TWebListInput.Items 2120
MidItems.TWebListInput.ItemsField 2120
MidItems.TWebListInput.Notification 2118
MidItems.TWebListInput.SetDataSet 2119
MidItems.TWebListInput.SetItems 2119
MidItems.TWebListInput.SetValues 2119
MidItems.TWebListInput.Values 2121
MidItems.TWebListInput.ValuesField 2121
MidItems.TWebRadioGroupInput 2122
MidItems.TWebRadioGroupInput.AddAttributes 2125
MidItems.TWebRadioGroupInput.Create 2125
MidItems.TWebRadioGroupInput.DisplayColumns 2127
MidItems.TWebRadioGroupInput.DisplayWidth 2127
MidItems.TWebRadioGroupInput.EventContent 2126
MidItems.TWebRadioGroupInput.FormatInputs 2126
MidItems.TWebRadioGroupInput.GetCheckIndex 2126
MidItems.TWebRadioGroupInput.ImplGetRowSetFieldAttributes 2126
MidItems.TWebRadioGroupInput.ReadOnly 2128
MidItems.TWebRadioGroupInput.RestoreFieldDefaults 2126
MidItems.TWebSelectOptionsInput 2128
MidItems.TWebSelectOptionsInput.AddAttributes 2131
MidItems.TWebSelectOptionsInput.Create 2131
MidItems.TWebSelectOptionsInput.DisplayRows 2132
MidItems.TWebSelectOptionsInput.EventContent 2132
MidItems.TWebSelectOptionsInput.FormatInputs 2132
MidItems.TWebSelectOptionsInput.GetSelectIndex 2132
MidItems.TWebStatus 2133
MidItems.TWebStatus.Caption 2137
MidItems.TWebStatus.CaptionAttributes 2137
MidItems.TWebStatus.CaptionPosition 2138
MidItems.TWebStatus.ControlContent 2135
MidItems.TWebStatus.Create 2135
MidItems.TWebStatus.DisplayWidth 2138
MidItems.TWebStatus.GetCaption 2136
MidItems.TWebStatus.Identifier 2136
MidItems.TWebStatus.ImplGetHTMLControlName 2136
MidItems.TWebStatusClass 2241
MidItems.TWebStatusClass type 2241
MidItems.TWebTextAreaInput 2138
MidItems.TWebTextAreaInput.AddAttributes 2142
MidItems.TWebTextAreaInput.ControlContent 2142

MidItems.TWebTextAreaInput.Create 2141
MidItems.TWebTextAreaInput.DisplayRows 2143
MidItems.TWebTextAreaInput.DisplayWidth 2144
MidItems.TWebTextAreaInput.EventContent 2142
MidItems.TWebTextAreaInput.ImplGetRowSetFieldAttributes 2143
MidItems.TWebTextAreaInput.ReadOnly 2144
MidItems.TWebTextAreaInput.RestoreFieldDefaults 2143
MidItems.TWebTextAreaInput.Wrap 2144
MidItems.TWebTextInput 2145
MidItems.TWebTextInput.AddAttributes 2148
MidItems.TWebTextInput.ControlContent 2148
MidItems.TWebTextInput.Create 2147
MidItems.TWebTextInput.Custom 2149
MidItems.TWebTextInput.DisplayWidth 2150
MidItems.TWebTextInput.ImplGetRowSetFieldAttributes 2149
MidItems.TWebTextInput.ReadOnly 2150
MidItems.TWebTextInput.RestoreFieldDefaults 2149
MidItems.TWebTextInput.Style 2150
MidItems.TWebTextInput.StyleRule 2151
MidItems.TXMLButton 2151
MidItems.TXMLButton.Create 2153
MidItems.TXMLButton.DefaultCaption 2156
MidItems.TXMLButton.Destroy 2153
MidItems.TXMLButton.GetCaption 2154
MidItems.TXMLButton.GetXMLBroker 2154
MidItems.TXMLButton.GetXMLUseParent 2154
MidItems.TXMLButton.Notification 2154
MidItems.TXMLButton.SetXMLBroker 2155
MidItems.TXMLButton.SetXMLUseParent 2155
MidItems.TXMLButton.XMLBroker 2156
MidItems.TXMLButton.XMLData 2156
MidItems.TXMLButton.XMLUseParent 2156
MidItems.TXMLData 2157
MidItems.TXMLData.Changed 2159
MidItems.TXMLData.ChangedXMLBroker 2159
MidItems.TXMLData.Create 2158
MidItems.TXMLData.GetXMLBroker 2159
MidItems.TXMLData.Notification 2159
MidItems.TXMLData.OnChange 2161
MidItems.TXMLData.Parent 2160
MidItems.TXMLData.SetXMLBroker 2160
MidItems.TXMLData.XMLBroker 2160
MidItems.TXMLDataParent 2161
MidItems.TXMLDataParent.ChangedUseParent 2163
MidItems.TXMLDataParent.ChangedXMLBroker 2163
MidItems.TXMLDataParent.GetXMLBroker 2163
MidItems.TXMLDataParent.ParentXMLBroker 2163
MidItems.TXMLDataParent.SetUseParent 2163
MidItems.TXMLDataParent.UseParent 2164
MidItems.TXMLDataSet 2164
MidItems.TXMLDataSet._AddRef 2167
MidItems.TXMLDataSet._Release 2167
MidItems.TXMLDataSet.ChangedDataSetField 2167
MidItems.TXMLDataSet.ChangedXMLBroker 2167
MidItems.TXMLDataSet.CompareXMLData 2167
MidItems.TXMLDataSet.ConnectionChange 2168
MidItems.TXMLDataSet.Create 2165
MidItems.TXMLDataSet.CreateDataSet 2168
MidItems.TXMLDataSet.DataSet 2170
MidItems.TXMLDataSet.DataSetField 2171
MidItems.TXMLDataSet.DataSetPath 2171
MidItems.TXMLDataSet.Destroy 2166
MidItems.TXMLDataSet.FindDataSet 2168
MidItems.TXMLDataSet.GetDataSet 2168
MidItems.TXMLDataSet.GetDataSetField 2169
MidItems.TXMLDataSet.GetDataSetPath 2169
MidItems.TXMLDataSet.Parent 2171
MidItems.TXMLDataSet.QueryInterface 2169
MidItems.TXMLDataSet.SetDataSetField 2169
MidItems.TXMLDataSet.SetXMLBroker 2169
MidItems.TXMLDataSet.WrapperDataChange 2170
MidItems.TXMLDataSetParent 2172
MidItems.TXMLDataSetParent.ChangedDataSetField 2174
MidItems.TXMLDataSetParent.ChangedUseParent 2174
MidItems.TXMLDataSetParent.ChangedXMLBroker 2174
MidItems.TXMLDataSetParent.GetDataSetField 2174
MidItems.TXMLDataSetParent.GetXMLBroker 2174
MidItems.TXMLDataSetParent.ParentDataSetField 2175

MidItems.TXMLDataSetParent.ParentXMLBroker 2175
MidItems.TXMLDataSetParent.ParentXMLDataSet 2175
MidItems.TXMLDataSetParent.SetUseParent 2175
MidItems.TXMLDataSetParent.UseParent 2176
MidItems.TXMLDisplayGroup 2176
MidItems.TXMLDisplayGroup.Create 2178
MidItems.TXMLDisplayGroup.CreateDefaultFields 2179
MidItems.TXMLDisplayGroup.Destroy 2178
MidItems.TXMLDisplayGroup.EnableValidateFields 2186
MidItems.TXMLDisplayGroup.FindField 2180
MidItems.TXMLDisplayGroup.GetDataSet 2180
MidItems.TXMLDisplayGroup.GetDataSetField 2180
MidItems.TXMLDisplayGroup.GetEnableValidateFields 2180
MidItems.TXMLDisplayGroup.GetFieldsList 2181
MidItems.TXMLDisplayGroup.GetIsMultipleRecordView 2181
MidItems.TXMLDisplayGroup.GetVisibleFields 2181
MidItems.TXMLDisplayGroup.GetXMLBroker 2181
MidItems.TXMLDisplayGroup.GetXMLDataSet 2182
MidItems.TXMLDisplayGroup.GetXMLDisplayName 2182
MidItems.TXMLDisplayGroup.GetXMLRowSetName 2182
MidItems.TXMLDisplayGroup.HasStatusField 2182
MidItems.TXMLDisplayGroup.ImplAddElements 2183
MidItems.TXMLDisplayGroup.ImplGetSubComponents 2183
MidItems.TXMLDisplayGroup.ImplIsMultipleRecordView 2183
MidItems.TXMLDisplayGroup.IsFieldInUse 2184
MidItems.TXMLDisplayGroup.Notification 2184
MidItems.TXMLDisplayGroup.SetDataSetField 2184
MidItems.TXMLDisplayGroup.SetEnableValidateFields 2185
MidItems.TXMLDisplayGroup.SetXMLBroker 2185
MidItems.TXMLDisplayGroup.ValidateDataSet 2185
MidItems.TXMLDisplayGroup.ValidateFields 2185
MidItems.TXMLDisplayGroup.VisibleFields 2186
MidItems.TXMLDisplayGroup.XMLBroker 2187
MidItems.TXMLDisplayGroup.XMLDataChange 2185
MidItems.TXMLDisplayGroup.XMLDataSet 2187
MidItems.TXMLDisplayGroup.XMLDataSetField 2188
MidItems.TXMLDisplayParent 2188
MidItems.TXMLDisplayParent.ChangedDisplayComponent 2190
MidItems.TXMLDisplayParent.ChangedUseParent 2190
MidItems.TXMLDisplayParent.Create 2189
MidItems.TXMLDisplayParent.GetDisplayComponent 2190
MidItems.TXMLDisplayParent.SetUseParent 2190
MidItems.TXMLDisplayParent.UseParent 2191
MidItems.TXMLDisplayReferenceButton 2191
MidItems.TXMLDisplayReferenceButton.Create 2193
MidItems.TXMLDisplayReferenceButton.Destroy 2193
MidItems.TXMLDisplayReferenceButton.GetDisplayComponent 2194
MidItems.TXMLDisplayReferenceButton.GetDisplayComponentParent 2194
MidItems.TXMLDisplayReferenceButton.GetXmlDisplayName 2194
MidItems.TXMLDisplayReferenceButton.Notification 2194
MidItems.TXMLDisplayReferenceButton.SetDisplayComponent 2195
MidItems.TXMLDisplayReferenceButton.SetDisplayComponentParent 2195
MidItems.TXMLDisplayReferenceButton.XMLComponent 2195
MidItems.TXMLDisplayReferenceButton.XMLDisplay 2196
MidItems.TXMLDisplayReferenceButton.XMLUseParent 2196
MidItems.TXMLDisplayReferenceGroup 2197
MidItems.TXMLDisplayReferenceGroup.ImplAddElements 2198
MidItems.TXMLDisplayReferenceGroup.XMLComponent 2199
MidProd 2255
MidProd namespace 2255
Classes 2257
Constants 2301
Functions 2296
Structs, Records, Enums 2297
Types 2298
MidProd.AddStdIncludes 2296
MidProd.AddStdIncludes function 2296
MidProd.PathInfoToRelativePath 2296
MidProd.PathInfoToRelativePath function 2296
MidProd.sXMLBroker 2301
MidProd.sXMLBroker constant 2301
MidProd.sXMLReadyVar 2301
MidProd.sXMLReadyVar constant 2301
MidProd.TBaseInetXPageProducer 2257

MidProd.TBaseInetXPageProducer.AddIncludes 2261
MidProd.TBaseInetXPageProducer.AddScriptComponents 2261
MidProd.TBaseInetXPageProducer.CanAddClass 2261
MidProd.TBaseInetXPageProducer.ContentFromStream 2262
MidProd.TBaseInetXPageProducer.Create 2259
MidProd.TBaseInetXPageProducer.CreatePageElements 2262
MidProd.TBaseInetXPageProducer.CreateScriptManager 2262
MidProd.TBaseInetXPageProducer.DeclareFunctions 2263
MidProd.TBaseInetXPageProducer.DeclareIncludes 2263
MidProd.TBaseInetXPageProducer.DeclareRowSets 2263
MidProd.TBaseInetXPageProducer.DeclareScript 2263
MidProd.TBaseInetXPageProducer.DeclareStyles 2263
MidProd.TBaseInetXPageProducer.DeclareWarnings 2264
MidProd.TBaseInetXPageProducer.DeclareXMLDocuments 2264
MidProd.TBaseInetXPageProducer.DeclareXMLIslands 2264
MidProd.TBaseInetXPageProducer.Destroy 2259
MidProd.TBaseInetXPageProducer.DoAfterGetContent 2264
MidProd.TBaseInetXPageProducer.DoAfterGetXMLData 2265
MidProd.TBaseInetXPageProducer.DoBeforeGetContent 2265
MidProd.TBaseInetXPageProducer.DoBeforeGetXMLData 2265
MidProd.TBaseInetXPageProducer.DoTagEvent 2265
MidProd.TBaseInetXPageProducer.EnableXMLIslands 2271
MidProd.TBaseInetXPageProducer.FindXMLBroker 2265
MidProd.TBaseInetXPageProducer.FindXMLBrokerName 2266
MidProd.TBaseInetXPageProducer.GeneratePageElements 2266
MidProd.TBaseInetXPageProducer.GetContentOptions 2267
MidProd.TBaseInetXPageProducer.GetDefaultTemplate 2267
MidProd.TBaseInetXPageProducer.GetRequiredTags 2267
MidProd.TBaseInetXPageProducer.GetScriptManager 2268
MidProd.TBaseInetXPageProducer.GetUseXMLIslands 2268
MidProd.TBaseInetXPageProducer.GetXMLData 2268
MidProd.TBaseInetXPageProducer.GetXMLDelta 2268
MidProd.TBaseInetXPageProducer.GetXMLErrors 2269
MidProd.TBaseInetXPageProducer.ImplCanAddClass 2269
MidProd.TBaseInetXPageProducer.IncludePathURL 2271
MidProd.TBaseInetXPageProducer.OnAfterGetContent 2273
MidProd.TBaseInetXPageProducer.OnAfterGetXMLData 2274
MidProd.TBaseInetXPageProducer.OnBeforeGetContent 2274
MidProd.TBaseInetXPageProducer.OnBeforeGetXMLData 2275
MidProd.TBaseInetXPageProducer.PageElements 2271
MidProd.TBaseInetXPageProducer.ScriptManager 2272
MidProd.TBaseInetXPageProducer.SetIncludePathURL 2269
MidProd.TBaseInetXPageProducer.SetStyles 2269
MidProd.TBaseInetXPageProducer.SetStylesFile 2270
MidProd.TBaseInetXPageProducer.SetWebContentOptions 2270
MidProd.TBaseInetXPageProducer.Styles 2272
MidProd.TBaseInetXPageProducer.StylesFile 2273
MidProd.TBaseMidasPageProducer 2298
MidProd.TBaseMidasPageProducer type 2298
MidProd.TCustomInetXPageProducer 2275
MidProd.TCustomInetXPageProducer.Content 2279
MidProd.TCustomInetXPageProducer.Create 2278
MidProd.TCustomInetXPageProducer.Destroy 2278
MidProd.TCustomInetXPageProducer.GetTemplateFileName 2280
MidProd.TCustomInetXPageProducer.GetTemplateStream 2280
MidProd.TCustomInetXPageProducer.HTMLDoc 2281
MidProd.TCustomInetXPageProducer.HTMLFile 2282
MidProd.TCustomInetXPageProducer.WebContent 2281
MidProd.TCustomMidasPageProducer 2298
MidProd.TCustomMidasPageProducer type 2298
MidProd.TInetXPageElements 2283
MidProd.TInetXPageElements.BodyContent 2285
MidProd.TInetXPageElements.FormsContent 2284
MidProd.TInetXPageElements.IncludesContent 2284
MidProd.TInetXPageElements.ScriptContent 2284
MidProd.TInetXPageElements.StylesContent 2284
MidProd.TInetXPageElements.WarningsContent 2284
MidProd.TInetXPageProducer 2285
MidProd.TInetXPageProducer.EnableXMLIslands 2288
MidProd.TInetXPageProducer.HTMLDoc 2288
MidProd.TInetXPageProducer.HTMLFile 2289
MidProd.TInetXPageProducer.IncludePathURL 2290
MidProd.TInetXPageProducer.OnAfterGetContent 2293
MidProd.TInetXPageProducer.OnAfterGetXMLData 2293
MidProd.TInetXPageProducer.OnBeforeGetContent 2294
MidProd.TInetXPageProducer.OnBeforeGetXMLData 2294

MidProd.TInetXPageProducer.ScriptEngine 2290
MidProd.TInetXPageProducer.Styles 2291
MidProd.TInetXPageProducer.StylesFile 2291
MidProd.TInetXPageProducer.WebPageItems 2292
MidProd.TMidasPageElements 2299
MidProd.TMidasPageElements type 2299
MidProd.TMidasPageProducer 2295
MidProd.TStdInclude 2297
MidProd.TStdInclude enumeration 2297
MidProd.TStdIncludes 2299
MidProd.TStdIncludes type 2299
MidProd.TXMLDataEvent 2299
MidProd.TXMLDataEvent type 2299
Minimized
TCustomRibbon class 3397
MinTabwidth
TRibbonTabItem class 3558
MinTruncatedWidth
TRibbonTabItem class 3559
MinValue
TRibbonSpinEdit class 3854
MMNotify
TMediaPlayer class 2318
moClient enumeration member 331
moConServer enumeration member 331
Mode
TIBBatchMove class 34
TIBBlobStream class 52
TMediaPlayer class 2350
moDevelopment enumeration member 331
Modified
TIBBlobStream class 53
TIBXSQLDA class 543
TIBXSQLVAR class 554
TOleContainer class 2567
TRibbonTabItem class 3559
Modify
TStringHash class 785
ModifyFirstName enumeration member 481
ModifyGroupId enumeration member 481
ModifyLastName enumeration member 481
ModifyMiddleName enumeration member 481
ModifyPassword enumeration member 481
ModifySQL
TIBCustomDataSet class 101
TIBDataSet class 128
TIBUpdateSQL class 636
ModifyUser
TIBSecurityService class 448
ModifyUserId enumeration member 481
moDocumentation enumeration member 331
moGuiTools enumeration member 331
moServer enumeration member 331
MouseDown
TCustomOutline class 2858
TCustomRibbon class 3384
TCustomRibbonGroup class 3417
TOffice2007Button class 3464
TOleContainer class 2545
TRibbonComboControl class 3692
TRibbonLargeSplitButton class 3750
TRibbonSmallSplitButton class 3794
TTimerSpeedButton class 3909
MouseInControl
TCustomRibbonComboBox class 3592
TOffice2007Button class 3467
TRibbonApplicationMenuButton class 3612
TRibbonComboControl class 3697
TRibbonGroupCollapsedControl class 3518
TRibbonGroupScrollButton class 3524
TTimerSpeedButton class 3910
MouseInSplit
TRibbonSplitMenuItem class 3902
MouseInSystemMenu
TCustomRibbon class 3385
MouseMove
TCustomRibbon class 3385
TCustomRibbonComboBox class 3588

TCustomRibbonGroup class 3417
TOffice2007Button class 3465
TPanningWindow class 697
TRibbonLargeSplitButton class 3750
TRibbonQATScrollButton class 3770
TRibbonSmallSplitButton class 3795
TRibbonSplitMenuItem class 3901
MouseUp
TCustomRibbon class 3385
TCustomRibbonGroup class 3417
TOffice2007Button class 3465
TRibbonLargeSplitButton class 3751
TRibbonMenuSmallButtonControl class 3764
TTimerSpeedButton class 3909
Move
TCustomImageList class 671
TCustomOutline class 2858
MovedCount
TIBBatchMove class 34
MoveGroup
TCustomRibbon class 3386
MoveKey
TRegistry class 3269
MoveNext
TKeyTipListEnumerator class 3597
TMenuItemEnumerator class 1425
MoveTo
TOutlineNode class 2949
mpCanEject enumeration member 2378
mpCanPlay enumeration member 2378
mpCanRecord enumeration member 2378
mpCanStep enumeration member 2378
MPlayer 2303
MPlayer namespace 2303
Classes 2305
Structs, Records, Enums 2376
Types 2384
MPlayer.EMCIDeviceError 2305
MPlayer.EMPNotify 2384
MPlayer.EMPNotify type 2384
MPlayer.EMPPostNotify 2384
MPlayer.EMPPostNotify type 2384
MPlayer.TButtonSet 2385
MPlayer.TButtonSet type 2385
MPlayer.TMediaDevice 2385
MPlayer.TMediaDevice type 2385
MPlayer.TMediaPlayer 2305
MPlayer.TMediaPlayer.Anchors 2332
MPlayer.TMediaPlayer.AutoButtonSet 2312
MPlayer.TMediaPlayer.AutoEnable 2333
MPlayer.TMediaPlayer.AutoOpen 2336
MPlayer.TMediaPlayer.AutoRewind 2337
MPlayer.TMediaPlayer.Back 2313
MPlayer.TMediaPlayer.Capabilities 2337
MPlayer.TMediaPlayer.Click 2313
MPlayer.TMediaPlayer.Close 2314
MPlayer.TMediaPlayer.ColoredButtons 2337
MPlayer.TMediaPlayer.Constraints 2338
MPlayer.TMediaPlayer.Create 2311
MPlayer.TMediaPlayer.Destroy 2311
MPlayer.TMediaPlayer.DeviceID 2338
MPlayer.TMediaPlayer.DeviceType 2339
MPlayer.TMediaPlayer.Display 2339
MPlayer.TMediaPlayer.DisplayRect 2340
MPlayer.TMediaPlayer.DoNotify 2315
MPlayer.TMediaPlayer.DoubleBuffered 2340
MPlayer.TMediaPlayer.Eject 2315
MPlayer.TMediaPlayer.Enabled 2342
MPlayer.TMediaPlayer.EnabledButtons 2343
MPlayer.TMediaPlayer.EndPos 2345
MPlayer.TMediaPlayer.Error 2346
MPlayer.TMediaPlayer.ErrorMessage 2346
MPlayer.TMediaPlayer.FileName 2347
MPlayer.TMediaPlayer.Frames 2349
MPlayer.TMediaPlayer.KeyDown 2317
MPlayer.TMediaPlayer.Length 2350
MPlayer.TMediaPlayer.Loaded 2317
MPlayer.TMediaPlayer.MMNotify 2318

MPlayer.TMediaPlayer.Mode 2350
MPlayer.TMediaPlayer.Next 2318
MPlayer.TMediaPlayer.Notification 2319
MPlayer.TMediaPlayer.Notify 2350
MPlayer.TMediaPlayer.NotifyValue 2351
MPlayer.TMediaPlayer.OnClick 2366
MPlayer.TMediaPlayer.OnContextPopup 2367
MPlayer.TMediaPlayer.OnEnter 2368
MPlayer.TMediaPlayer.OnExit 2370
MPlayer.TMediaPlayer.OnMouseEnter 2374
MPlayer.TMediaPlayer.OnMouseLeave 2374
MPlayer.TMediaPlayer.OnNotify 2375
MPlayer.TMediaPlayer.OnPostClick 2375
MPlayer.TMediaPlayer.Open 2319
MPlayer.TMediaPlayer.Paint 2322
MPlayer.TMediaPlayer.ParentDoubleBuffered 2352
MPlayer.TMediaPlayer.ParentShowHint 2352
MPlayer.TMediaPlayer.Pause 2322
MPlayer.TMediaPlayer.PauseOnly 2323
MPlayer.TMediaPlayer.Play 2323
MPlayer.TMediaPlayer.PopupMenu 2352
MPlayer.TMediaPlayer.Position 2357
MPlayer.TMediaPlayer.PostClick 2326
MPlayer.TMediaPlayer.Previous 2327
MPlayer.TMediaPlayer.Resume 2327
MPlayer.TMediaPlayer.Rewind 2328
MPlayer.TMediaPlayer.Save 2328
MPlayer.TMediaPlayer.Shareable 2358
MPlayer.TMediaPlayer.ShowHint 2358
MPlayer.TMediaPlayer.Start 2360
MPlayer.TMediaPlayer.StartPos 2360
MPlayer.TMediaPlayer.StartRecording 2329
MPlayer.TMediaPlayer.Step 2330
MPlayer.TMediaPlayer.Stop 2330
MPlayer.TMediaPlayer.TabOrder 2361
MPlayer.TMediaPlayer.TabStop 2361
MPlayer.TMediaPlayer.TimeFormat 2362
MPlayer.TMediaPlayer.TrackLength 2362
MPlayer.TMediaPlayer.TrackPosition 2362
MPlayer.TMediaPlayer.Tracks 2363
MPlayer.TMediaPlayer.Updated 2331
MPlayer.TMediaPlayer.Visible 2363
MPlayer.TMediaPlayer.VisibleButtons 2365
MPlayer.TMediaPlayer.Wait 2365
MPlayer.TMPBtnType 2376
MPlayer.TMPBtnType enumeration 2376
MPlayer.TMPButton 2377
MPlayer.TMPButton record 2377
MPlayer.TMPDevCaps 2378
MPlayer.TMPDevCaps enumeration 2378
MPlayer.TMPDevCapsSet 2385
MPlayer.TMPDevCapsSet type 2385
MPlayer.TMPDeviceTypes 2378
MPlayer.TMPDeviceTypes enumeration 2378
MPlayer.TMPGlyph 2380
MPlayer.TMPGlyph enumeration 2380
MPlayer.TMPModes 2380
MPlayer.TMPModes enumeration 2380
MPlayer.TMPNotifyValues 2381
MPlayer.TMPNotifyValues enumeration 2381
MPlayer.TMPTimeFormats 2382
MPlayer.TMPTimeFormats enumeration 2382
mpNotReady enumeration member 2380
mpOpen enumeration member 2380
mpPaused enumeration member 2380
mpPlaying enumeration member 2380
mpRecording enumeration member 2380
mpSeeking enumeration member 2380
mpStopped enumeration member 2380
mpUsesWindow enumeration member 2378
MsgFilePath
TIBSetup class 320
MsgToInvContext
IOPConvert interface 2735
TOPToSoapDomConvert class 2754
mtApplicationMenu enumeration member 3927
mtCollapsed enumeration member 3927
mtControl enumeration member 3927

mtDialogButton enumeration member 3927
mtGroup enumeration member 3927
Mtsobj 2387
Mtsobj namespace 2387
Classes 2388
Mtsobj.TMtsAutoObject 2388
Mtsobj.TMtsAutoObject.Activate 2389
Mtsobj.TMtsAutoObject.CanBePooled 2390
Mtsobj.TMtsAutoObject.Deactivate 2390
Mtsobj.TMtsAutoObject.DisableCommit 2390
Mtsobj.TMtsAutoObject.EnableCommit 2391
Mtsobj.TMtsAutoObject.IsCallerInRole 2391
Mtsobj.TMtsAutoObject.IsInTransaction 2392
Mtsobj.TMtsAutoObject.IsSecurityEnabled 2392
Mtsobj.TMtsAutoObject.ObjectContext 2394
Mtsobj.TMtsAutoObject.OnActivate 2393
Mtsobj.TMtsAutoObject.OnDeactivate 2393
Mtsobj.TMtsAutoObject.Pooled 2394
Mtsobj.TMtsAutoObject.SetAbort 2393
Mtsobj.TMtsAutoObject.SetComplete 2393
MtsRdm 2397
MtsRdm namespace 2397
Classes 2398
MtsRdm.TMtsDataModule 2398
MtsRdm.TMtsDataModule.Activate 2400
MtsRdm.TMtsDataModule.AS_ApplyUpdates 2401
MtsRdm.TMtsDataModule.AS_DataRequest 2401
MtsRdm.TMtsDataModule.AS_Execute 2402
MtsRdm.TMtsDataModule.AS_GetParams 2403
MtsRdm.TMtsDataModule.AS_GetProviderNames 2404
MtsRdm.TMtsDataModule.AS_GetRecords 2404
MtsRdm.TMtsDataModule.AS_RowRequest 2405
MtsRdm.TMtsDataModule.AutoComplete 2411
MtsRdm.TMtsDataModule.CallComplete 2406
MtsRdm.TMtsDataModule.CanBePooled 2406
MtsRdm.TMtsDataModule.Create 2399
MtsRdm.TMtsDataModule.Deactivate 2406
MtsRdm.TMtsDataModule.DisableCommit 2407
MtsRdm.TMtsDataModule.EnableCommit 2407
MtsRdm.TMtsDataModule.IsCallerInRole 2408
MtsRdm.TMtsDataModule.IsInTransaction 2408
MtsRdm.TMtsDataModule.IsSecurityEnabled 2409
MtsRdm.TMtsDataModule.ObjectContext 2411
MtsRdm.TMtsDataModule.OnActivate 2412
MtsRdm.TMtsDataModule.OnDeactivate 2412
MtsRdm.TMtsDataModule.Pooled 2412
MtsRdm.TMtsDataModule.SetAbort 2409
MtsRdm.TMtsDataModule.SetComplete 2410
mtTabs enumeration member 3927
Mtx 2415
Mtx namespace 2415
Constants 2442
Functions 2437
Interfaces 2417
Mtx.CLASS_SharedPropertyGroupManager 2442
Mtx.CLASS_SharedPropertyGroupManager constant 2442
Mtx.CLASS_TransactionContext 2442
Mtx.CLASS_TransactionContext constant 2442
Mtx.CLASS_TransactionContextEx 2443
Mtx.CLASS_TransactionContextEx constant 2443
Mtx.CreateSharedPropertyGroup 2437
Mtx.CreateSharedPropertyGroup function 2437
Mtx.CreateSharedPropertyGroupManager 2438
Mtx.CreateSharedPropertyGroupManager function 2438
Mtx.CreateTransactionContext 2438
Mtx.CreateTransactionContext function 2438
Mtx.CreateTransactionContextEx 2439
Mtx.CreateTransactionContextEx function 2439
Mtx.GetObjectContext 2440
Mtx.GetObjectContext function 2440
Mtx.IEnumNames 2417
Mtx.IEnumNames.Clone 2418
Mtx.IEnumNames.Next 2418
Mtx.IEnumNames.Reset 2418
Mtx.IEnumNames.Skip 2418
Mtx.IGetContextProperties 2419
Mtx.IGetContextProperties.Count 2419
Mtx.IGetContextProperties.EnumNames 2420

Mtx.IGetContextProperties.GetProperty 2420
Mtx.IObjectContext 2420
Mtx.IObjectContext.CreateInstance 2421
Mtx.IObjectContext.DisableCommit 2421
Mtx.IObjectContext.EnableCommit 2422
Mtx.IObjectContext.IsCallerInRole 2422
Mtx.IObjectContext.IsInTransaction 2422
Mtx.IObjectContext.IsSecurityEnabled 2422
Mtx.IObjectContext.SetAbort 2422
Mtx.IObjectContext.SetComplete 2423
Mtx.IObjectContextActivity 2423
Mtx.IObjectContextActivity.GetActivityId 2424
Mtx.IObjectControl 2424
Mtx.IObjectControl.Activate 2425
Mtx.IObjectControl.CanBePooled 2425
Mtx.IObjectControl.Deactivate 2425
Mtx.ISecurityProperty 2425
Mtx.ISecurityProperty.GetDirectCallerSID 2426
Mtx.ISecurityProperty.GetDirectCreatorSID 2426
Mtx.ISecurityProperty.GetOriginalCallerSID 2426
Mtx.ISecurityProperty.GetOriginalCreatorSID 2427
Mtx.ISecurityProperty.ReleaseSID 2427
Mtx.ISharedProperty 2427
Mtx.ISharedProperty.Get_Value 2428
Mtx.ISharedProperty.Set_Value 2428
Mtx.ISharedProperty.Value 2429
Mtx.ISharedPropertyGroup 2429
Mtx.ISharedPropertyGroup.CreateProperty 2430
Mtx.ISharedPropertyGroup.CreatePropertyByPosition 2430
Mtx.ISharedPropertyGroup.Get_PropertyByName 2430
Mtx.ISharedPropertyGroup.Get_PropertyByPosition 2430
Mtx.ISharedPropertyGroup.PropertyByName 2431
Mtx.ISharedPropertyGroup.PropertyByPosition 2431
Mtx.ISharedPropertyGroupManager 2431
Mtx.ISharedPropertyGroupManager._NewEnum 2433
Mtx.ISharedPropertyGroupManager.CreatePropertyGroup 2432
Mtx.ISharedPropertyGroupManager.Get__NewEnum 2432
Mtx.ISharedPropertyGroupManager.Get_Group 2433
Mtx.ISharedPropertyGroupManager.Group 2433
Mtx.ITransactionContext 2433
Mtx.ITransactionContext.Abort 2434
Mtx.ITransactionContext.Commit 2434
Mtx.ITransactionContext.CreateInstance 2435
Mtx.ITransactionContextEx 2435
Mtx.ITransactionContextEx.Abort 2436
Mtx.ITransactionContextEx.Commit 2436
Mtx.ITransactionContextEx.CreateInstance 2436
Mtx.LockMethod 2443
Mtx.LockMethod constant 2443
Mtx.LockSetGet 2443
Mtx.LockSetGet constant 2443
Mtx.mtsErrCtxAborted 2444
Mtx.mtsErrCtxAborted constant 2444
Mtx.mtsErrCtxAborting 2444
Mtx.mtsErrCtxAborting constant 2444
Mtx.mtsErrCtxActivityTimeout 2444
Mtx.mtsErrCtxActivityTimeout constant 2444
Mtx.mtsErrCtxNoContext 2445
Mtx.mtsErrCtxNoContext constant 2445
Mtx.mtsErrCtxNoSecurity 2445
Mtx.mtsErrCtxNoSecurity constant 2445
Mtx.mtsErrCtxNotRegistered 2445
Mtx.mtsErrCtxNotRegistered constant 2445
Mtx.mtsErrCtxOldReference 2446
Mtx.mtsErrCtxOldReference constant 2446
Mtx.mtsErrCtxRoleNotFound 2446
Mtx.mtsErrCtxRoleNotFound constant 2446
Mtx.mtsErrCtxTMNotAvailable 2446
Mtx.mtsErrCtxTMNotAvailable constant 2446
Mtx.mtsErrCtxWrongThread 2447
Mtx.mtsErrCtxWrongThread constant 2447
Mtx.Process 2447
Mtx.Process constant 2447
Mtx.SafeRef 2440
Mtx.SafeRef function 2440
Mtx.Standard 2447
Mtx.Standard constant 2447
MultiDatabase

TLimboTransactionInfo class 472
MultiRefObject
TSOAPDomConv class 2774
MustUnderstand
TSOAPHeader class 912

N

Name
ITypeCode interface 2819
TIBXSQLVAR class 554
TRibbonPage class 3543
TUpdateTree class 3211
NamedPipe enumeration member 479
Names
TIBXSQLDA class 543
NativeHandle
TIBEvents class 264
NativeToXS
TRemotableXS class 893
NCPaint
TCustomRibbonQuickAccessPopupToolbar class 3441
TRibbonActionPopupMenu class 3941
TRibbonApplicationPopupMenu class 3967
TRibbonQATCustomizeMenu class 3970
TRibbonStyleMainPopupMenu class 3976
NewBitmap
TJPEGImage class 1004
NewBottomLine
TMenuItem class 1386
NewImage
TJPEGImage class 1004
NewInserted
TOleContainer class 2567
NewInstance
TInvokableClass class 838
TRIO class 4062
NewPage
TPrinter class 3047
NewPopup
TRibbonApplicationMenuBar class 3947
NewRequest
TISAPIFactory class 964
NewResponse
TISAPIFactory class 964
NewTopLine
TMenuItem class 1387
NewValuesModified
TPacketDataSet class 3195
NewWidth
TCustomTabData class 3458
NewXMLDocument
TOPToSoapDomConvert class 2754
Next
IEnumNames interface 2418
TIBSQL class 527
TMediaPlayer class 2318
NextColumn
TFormLayout class 1920
NodeIsNULL
TSOAPDomConv class 2774
NoGarbageCollection enumeration member 477
NoGlobalAction enumeration member 483
NonExistent
IStub interface 2814
NonTransportable enumeration member 477
NoOfAttachments
TDatabaseInfo class 405
NoOfDatabases
TDatabaseInfo class 406
NoReserve
TIBDatabaseInfo class 251
NoShadow enumeration member 480
Notification
TCustomRibbon class 3386
TCustomRibbonComboBox class 3588
TCustomRibbonGroup class 3417
TCustomRibbonPage class 3433
TCustomRibbonQuickAccessToolbar class 3449

TDataSetProvider class 3157
TDispatchConnection class 1294
TIBBatchMove class 29
TIBDataBase class 187
TIBEvents class 261
TIBExtract class 280
TIBTransaction class 212
TMediaPlayer class 2319
TMenu class 1341
TMenuItem class 1387
TQueryButton class 1958
TRibbonApplicationMenuBar class 3947
TRibbonComboControl class 3693
TRibbonCustomControl class 3704
TRibbonGroupCollapsedControl class 3512
TRibbonSpinButton class 3799
TSharedConnection class 1313
TWebDisplay class 2097
TWebDisplayReferenceGroup class 2101
TWebListInput class 2118
TXMLButton class 2154
TXMLData class 2159
TXMLDisplayGroup class 2184
TXMLDisplayReferenceButton class 2194
Notify
TMediaPlayer class 2350
TRibbonTabs class 3562
TStaticListItems class 1065
NotifyValue
TMediaPlayer class 2351
NoValidityCheck enumeration member 480
nsClientReceive enumeration member 2784
nsClientSend enumeration member 2784
nsServerReceive enumeration member 2784
nsServerSend enumeration member 2784
NumbersOnly
TRibbonComboBox class 3636
NumBuffers
TIBDatabaseInfo class 252
nvAborted enumeration member 2381
nvFailure enumeration member 2381
nvSuccessful enumeration member 2381
nvSuperseded enumeration member 2381

O

oaAdd enumeration member 2957
oaAddChild enumeration member 2957
oaInsert enumeration member 2957
obClose enumeration member 2958
ObjAuto 2449
ObjAuto namespace 2449
Constants 2463
Functions 2453
Interfaces 2451
Structs, Records, Enums 2456
Types 2459
ObjAuto.CreateMethodPointer 2453
ObjAuto.CreateMethodPointer function 2453
ObjAuto.GetInvokeInstance 2454
ObjAuto.GetInvokeInstance function 2454
ObjAuto.GetMethodInfo 2454
ObjAuto.GetMethodInfo function 2454
ObjAuto.GetMethods 2454
ObjAuto.GetMethods function 2454
ObjAuto.IMethodHandler 2451
ObjAuto.IMethodHandler.Execute 2452
ObjAuto.IMethodHandler.InstanceToVariant 2452
ObjAuto.ObjectInvoke 2455
ObjAuto.ObjectInvoke function 2455
ObjAuto.paEAX 2463
ObjAuto.paEAX constant 2463
ObjAuto.paECX 2463
ObjAuto.paECX constant 2463
ObjAuto.paEDX 2464
ObjAuto.paEDX constant 2464
ObjAuto.paStack 2464
ObjAuto.paStack constant 2464
ObjAuto.PMethodInfoHeader 2459

ObjAuto.PMethodInfoHeader type 2459
ObjAuto.PParameters 2459
ObjAuto.PParameters type 2459
ObjAuto.PParamInfo 2460
ObjAuto.PParamInfo type 2460
ObjAuto.PPointer 2460
ObjAuto.PPointer type 2460
ObjAuto.PReturnInfo 2460
ObjAuto.PReturnInfo type 2460
ObjAuto.PWord 2461
ObjAuto.PWord type 2461
ObjAuto.ReleaseMethodPointer 2455
ObjAuto.ReleaseMethodPointer function 2455
ObjAuto.TCallingConvention 2456
ObjAuto.TCallingConvention enumeration 2456
ObjAuto.TDynamicInvokeEvent 2461
ObjAuto.TDynamicInvokeEvent type 2461
ObjAuto.TMethodInfoArray 2461
ObjAuto.TMethodInfoArray type 2461
ObjAuto.TMethodInfoHeader 2456
ObjAuto.TMethodInfoHeader record 2456
ObjAuto.TParameters 2457
ObjAuto.TParameters record 2457
ObjAuto.TParamFlags 2462
ObjAuto.TParamFlags type 2462
ObjAuto.TParamInfo 2457
ObjAuto.TParamInfo record 2457
ObjAuto.TReturnInfo 2458
ObjAuto.TReturnInfo record 2458
ObjBrkr 2465
ObjBrkr namespace 2465
Classes 2466
ObjBrkr.EBrokerException 2466
ObjBrkr.TServerCollection 2466
ObjBrkr.TServerCollection.Create 2467
ObjBrkr.TServerCollection.FindServer 2468
ObjBrkr.TServerCollection.GetBalancedName 2468
ObjBrkr.TServerCollection.GetNextName 2469
ObjBrkr.TServerCollection.Items 2470
ObjBrkr.TServerItem 2470
ObjBrkr.TServerItem.ComputerName 2472
ObjBrkr.TServerItem.Create 2471
ObjBrkr.TServerItem.Enabled 2473
ObjBrkr.TServerItem.GetDisplayName 2472
ObjBrkr.TServerItem.HasFailed 2474
ObjBrkr.TServerItem.Port 2474
ObjBrkr.TSimpleObjectBroker 2475
ObjBrkr.TSimpleObjectBroker.Create 2476
ObjBrkr.TSimpleObjectBroker.Destroy 2476
ObjBrkr.TSimpleObjectBroker.GetComputerForGUID 2477
ObjBrkr.TSimpleObjectBroker.GetComputerForProgID 2477
ObjBrkr.TSimpleObjectBroker.GetPortForComputer 2478
ObjBrkr.TSimpleObjectBroker.LoadBalanced 2480
ObjBrkr.TSimpleObjectBroker.LoadFromStream 2479
ObjBrkr.TSimpleObjectBroker.SaveToStream 2479
ObjBrkr.TSimpleObjectBroker.ServerData 2481
ObjBrkr.TSimpleObjectBroker.Servers 2481
ObjBrkr.TSimpleObjectBroker.SetConnectStatus 2479
ObjComAuto 2483
ObjComAuto namespace 2483
Classes 2484
Functions 2489
Structs, Records, Enums 2491
Types 2493
ObjComAuto.CreateMethodPointer 2489
ObjComAuto.CreateMethodPointer function 2489
ObjComAuto.ReleaseMethodPointer 2490
ObjComAuto.ReleaseMethodPointer function 2490
ObjComAuto.TDispatchInfo 2491
ObjComAuto.TDispatchInfo record 2491
ObjComAuto.TDispatchInfos 2493
ObjComAuto.TDispatchInfos type 2493
ObjComAuto.TDispatchKind 2491
ObjComAuto.TDispatchKind enumeration 2491
ObjComAuto.TObjectDispatch 2484
ObjComAuto.TObjectDispatch.Create 2485
ObjComAuto.TObjectDispatch.Destroy 2485
ObjComAuto.TObjectDispatch.GetIDsOfNames 2486

ObjComAuto.TObjectDispatch.GetMethodInfo 2486
ObjComAuto.TObjectDispatch.GetObjectDispatch 2486
ObjComAuto.TObjectDispatch.GetPropInfo 2487
ObjComAuto.TObjectDispatch.GetTypeInfo 2487
ObjComAuto.TObjectDispatch.GetTypeInfoCount 2487
ObjComAuto.TObjectDispatch.Instance 2488
ObjComAuto.TObjectDispatch.Invoke 2488
ObjectBroker
TDCOMConnection class 1288
TDispatchConnection class 1300
ObjectContext
TMtsAutoObject class 2394
TMtsDataModule class 2411
ObjectMemberNoShow
TSOAPDomConv class 2775
ObjectPropertiesDialog
TOleContainer class 2546
ObjectToSOAP
TRemotable class 871
TRemotableXS class 894
TSOAPAttachment class 900
TSOAPHeader class 910
ObjectToString
IORB interface 2808
ObjectVerbs
TOleContainer class 2568
ObjectView
TIBCustomDataSet class 101
ObjInstanceToSOAP
IObjConverter interface 922
TSOAPDomConv class 2775
ObjIsReady
IBOA interface 2795
obLeaf enumeration member 2958
obMinus enumeration member 2958
obOpen enumeration member 2958
obPlus enumeration member 2958
ocDefault enumeration member 943
ocGreater enumeration member 2958
ocInvalid enumeration member 2958
ocLess enumeration member 2958
ocMultiRef enumeration member 943
ocNoMultiRef enumeration member 943
ocoDontPrefixNode enumeration member 939
ocoDontPutTypeAttr enumeration member 939
ocoDontSerializeProps enumeration member 939
ocSame enumeration member 2958
ODSMajorVersion
TIBDatabaseInfo class 252
ODSMinorVersion
TIBDatabaseInfo class 252
OldMetadataDesc enumeration member 477
OldStreamFormat
TOleContainer class 2568
OldWidth
TRibbonGroupCollapsedControl class 3518
OleAuto 2495
OleAuto namespace 2495
Classes 2497
Constants 2525
Functions 2513
Structs, Records, Enums 2521
Types 2523
Variables 2524
OleAuto.Automation 2524
OleAuto.Automation variable 2524
OleAuto.ClassIDToProgID 2513
OleAuto.ClassIDToProgID function 2513
OleAuto.ClassIDToString 2513
OleAuto.ClassIDToString function 2513
OleAuto.CreateOleObject 2514
OleAuto.CreateOleObject function 2514
OleAuto.DispInvoke 2515
OleAuto.DispInvoke function 2515
OleAuto.DispInvokeError 2516
OleAuto.DispInvokeError function 2516
OleAuto.DllCanUnloadNow 2516
OleAuto.DllCanUnloadNow function 2516

OleAuto.DllGetClassObject 2516
OleAuto.DllGetClassObject function 2516
OleAuto.DllRegisterServer 2517
OleAuto.DllRegisterServer function 2517
OleAuto.DllUnregisterServer 2517
OleAuto.DllUnregisterServer function 2517
OleAuto.EOleError 2497
OleAuto.EOleException 2497
OleAuto.EOleException.Create 2498
OleAuto.EOleException.ErrorCode 2498
OleAuto.EOleException.HelpFile 2499
OleAuto.EOleException.Source 2499
OleAuto.EOleSysError 2499
OleAuto.EOleSysError.Create 2500
OleAuto.EOleSysError.ErrorCode 2500
OleAuto.GetActiveOleObject 2517
OleAuto.GetActiveOleObject function 2517
OleAuto.MaxDispArgs 2525
OleAuto.MaxDispArgs constant 2525
OleAuto.OleCheck 2518
OleAuto.OleCheck function 2518
OleAuto.OleError 2518
OleAuto.OleError function 2518
OleAuto.ProgIDToClassID 2518
OleAuto.ProgIDToClassID function 2518
OleAuto.StringToClassID 2519
OleAuto.StringToClassID function 2519
OleAuto.TAutoClass 2523
OleAuto.TAutoClass type 2523
OleAuto.TAutoClassInfo 2521
OleAuto.TAutoClassInfo record 2521
OleAuto.TAutoClassInstancing 2521
OleAuto.TAutoClassInstancing enumeration 2521
OleAuto.TAutoDispatch 2500
OleAuto.TAutoDispatch.AddRef 2502
OleAuto.TAutoDispatch.AutoObject 2504
OleAuto.TAutoDispatch.Create 2501
OleAuto.TAutoDispatch.GetAutoObject 2502
OleAuto.TAutoDispatch.GetIDsOfNames 2502
OleAuto.TAutoDispatch.GetTypeInfo 2502
OleAuto.TAutoDispatch.GetTypeInfoCount 2502
OleAuto.TAutoDispatch.Invoke 2503
OleAuto.TAutoDispatch.QueryInterface 2503
OleAuto.TAutoDispatch.Release 2503
OleAuto.TAutomation 2504
OleAuto.TAutomation.AutoObjectCount 2506
OleAuto.TAutomation.Create 2505
OleAuto.TAutomation.Destroy 2505
OleAuto.TAutomation.IsInprocServer 2506
OleAuto.TAutomation.OnLastRelease 2507
OleAuto.TAutomation.RegisterClass 2505
OleAuto.TAutomation.StartMode 2507
OleAuto.TAutomation.UpdateRegistry 2506
OleAuto.TAutoObject 2507
OleAuto.TAutoObject.AddRef 2509
OleAuto.TAutoObject.AutoDispatch 2510
OleAuto.TAutoObject.Create 2508
OleAuto.TAutoObject.CreateAutoDispatch 2510
OleAuto.TAutoObject.Destroy 2508
OleAuto.TAutoObject.GetExceptionInfo 2510
OleAuto.TAutoObject.OleObject 2511
OleAuto.TAutoObject.RefCount 2511
OleAuto.TAutoObject.Release 2510
OleAuto.TLastReleaseEvent 2523
OleAuto.TLastReleaseEvent type 2523
OleAuto.TRegistryClass 2511
OleAuto.TRegistryClass.Create 2512
OleAuto.TRegistryClass.Destroy 2512
OleAuto.TRegistryClass.UpdateRegistry 2512
OleAuto.TStartMode 2522
OleAuto.TStartMode enumeration 2522
OleAuto.VarFromInterface 2519
OleAuto.VarFromInterface function 2519
OleAuto.VarToAutoObject 2519
OleAuto.VarToAutoObject function 2519
OleAuto.VarToInterface 2520
OleAuto.VarToInterface function 2520
OleClassName

TOleContainer class 2569
OleControlSite_TranslateAccelerator
TOleControl class 2670
OleCtnrs 2527
OleCtnrs namespace 2527
Classes 2529
Constants 2623
Functions 2617
Interfaces 2615
Structs, Records, Enums 2618
Types 2622
OleCtnrs.DestroyMetaPict 2617
OleCtnrs.DestroyMetaPict function 2617
OleCtnrs.IVCLFrameForm 2615
OleCtnrs.IVCLFrameForm.AddContainer 2616
OleCtnrs.IVCLFrameForm.ClearBorderSpace 2616
OleCtnrs.IVCLFrameForm.Form 2616
OleCtnrs.IVCLFrameForm.RemoveContainer 2616
OleCtnrs.ovDiscardUndoState 2623
OleCtnrs.ovDiscardUndoState constant 2623
OleCtnrs.ovHide 2624
OleCtnrs.ovHide constant 2624
OleCtnrs.ovInPlaceActivate 2624
OleCtnrs.ovInPlaceActivate constant 2624
OleCtnrs.ovOpen 2625
OleCtnrs.ovOpen constant 2625
OleCtnrs.ovPrimary 2625
OleCtnrs.ovPrimary constant 2625
OleCtnrs.ovShow 2626
OleCtnrs.ovShow constant 2626
OleCtnrs.ovUIActivate 2627
OleCtnrs.ovUIActivate constant 2627
OleCtnrs.TAutoActivate 2618
OleCtnrs.TAutoActivate enumeration 2618
OleCtnrs.TCreateInfo 2618
OleCtnrs.TCreateInfo record 2618
OleCtnrs.TCreateType 2619
OleCtnrs.TCreateType enumeration 2619
OleCtnrs.TObjectMoveEvent 2622
OleCtnrs.TObjectMoveEvent type 2622
OleCtnrs.TObjectState 2620
OleCtnrs.TObjectState enumeration 2620
OleCtnrs.TOleContainer 2529
OleCtnrs.TOleContainer._AddRef 2534
OleCtnrs.TOleContainer._Release 2535
OleCtnrs.TOleContainer.Align 2552
OleCtnrs.TOleContainer.AllowActiveDoc 2556
OleCtnrs.TOleContainer.AllowInPlace 2556
OleCtnrs.TOleContainer.Anchors 2557
OleCtnrs.TOleContainer.AutoActivate 2557
OleCtnrs.TOleContainer.AutoVerbMenu 2558
OleCtnrs.TOleContainer.BorderStyle 2558
OleCtnrs.TOleContainer.CanPaste 2558
OleCtnrs.TOleContainer.Caption 2559
OleCtnrs.TOleContainer.Changed 2535
OleCtnrs.TOleContainer.ChangeIconDialog 2535
OleCtnrs.TOleContainer.Close 2536
OleCtnrs.TOleContainer.Color 2561
OleCtnrs.TOleContainer.Constraints 2562
OleCtnrs.TOleContainer.Copy 2536
OleCtnrs.TOleContainer.CopyOnSave 2563
OleCtnrs.TOleContainer.Create 2532
OleCtnrs.TOleContainer.CreateLinkToFile 2537
OleCtnrs.TOleContainer.CreateObject 2537
OleCtnrs.TOleContainer.CreateObjectFromFile 2538
OleCtnrs.TOleContainer.CreateObjectFromInfo 2539
OleCtnrs.TOleContainer.CreateParams 2540
OleCtnrs.TOleContainer.Ctl3D 2563
OleCtnrs.TOleContainer.DblClick 2540
OleCtnrs.TOleContainer.DefineProperties 2541
OleCtnrs.TOleContainer.Destroy 2533
OleCtnrs.TOleContainer.DestroyObject 2541
OleCtnrs.TOleContainer.DoEnter 2542
OleCtnrs.TOleContainer.DoVerb 2542
OleCtnrs.TOleContainer.DragCursor 2564
OleCtnrs.TOleContainer.DragMode 2564
OleCtnrs.TOleContainer.Enabled 2566
OleCtnrs.TOleContainer.GetIconMetaPict 2542

OleCtnrs.TOleContainer.GetPopupMenu 2543
OleCtnrs.TOleContainer.Iconic 2566
OleCtnrs.TOleContainer.InsertObjectDialog 2543
OleCtnrs.TOleContainer.KeyDown 2543
OleCtnrs.TOleContainer.Linked 2567
OleCtnrs.TOleContainer.LoadFromFile 2544
OleCtnrs.TOleContainer.LoadFromStream 2545
OleCtnrs.TOleContainer.Modified 2567
OleCtnrs.TOleContainer.MouseDown 2545
OleCtnrs.TOleContainer.NewInserted 2567
OleCtnrs.TOleContainer.ObjectPropertiesDialog 2546
OleCtnrs.TOleContainer.ObjectVerbs 2568
OleCtnrs.TOleContainer.OldStreamFormat 2568
OleCtnrs.TOleContainer.OleClassName 2569
OleCtnrs.TOleContainer.OleObject 2569
OleCtnrs.TOleContainer.OleObjectInterface 2569
OleCtnrs.TOleContainer.OnActivate 2583
OleCtnrs.TOleContainer.OnDeactivate 2583
OleCtnrs.TOleContainer.OnDragDrop 2584
OleCtnrs.TOleContainer.OnDragOver 2587
OleCtnrs.TOleContainer.OnEndDrag 2590
OleCtnrs.TOleContainer.OnEnter 2591
OleCtnrs.TOleContainer.OnExit 2593
OleCtnrs.TOleContainer.OnKeyDown 2597
OleCtnrs.TOleContainer.OnKeyPress 2601
OleCtnrs.TOleContainer.OnKeyUp 2602
OleCtnrs.TOleContainer.OnMouseActivate 2604
OleCtnrs.TOleContainer.OnMouseDown 2605
OleCtnrs.TOleContainer.OnMouseMove 2607
OleCtnrs.TOleContainer.OnMouseUp 2609
OleCtnrs.TOleContainer.OnObjectMove 2611
OleCtnrs.TOleContainer.OnResize 2612
OleCtnrs.TOleContainer.OnStartDrag 2612
OleCtnrs.TOleContainer.Paint 2546
OleCtnrs.TOleContainer.ParentColor 2570
OleCtnrs.TOleContainer.ParentCtl3D 2570
OleCtnrs.TOleContainer.ParentShowHint 2571
OleCtnrs.TOleContainer.Paste 2546
OleCtnrs.TOleContainer.PasteSpecialDialog 2547
OleCtnrs.TOleContainer.PopupMenu 2571
OleCtnrs.TOleContainer.PrimaryVerb 2576
OleCtnrs.TOleContainer.QueryInterface 2547
OleCtnrs.TOleContainer.Run 2548
OleCtnrs.TOleContainer.SaveAsDocument 2548
OleCtnrs.TOleContainer.SaveToFile 2550
OleCtnrs.TOleContainer.SaveToStream 2550
OleCtnrs.TOleContainer.ShowHint 2577
OleCtnrs.TOleContainer.SizeMode 2578
OleCtnrs.TOleContainer.SourceDoc 2579
OleCtnrs.TOleContainer.State 2579
OleCtnrs.TOleContainer.StorageInterface 2579
OleCtnrs.TOleContainer.TabOrder 2580
OleCtnrs.TOleContainer.TabStop 2580
OleCtnrs.TOleContainer.UpdateObject 2551
OleCtnrs.TOleContainer.UpdateVerbs 2551
OleCtnrs.TOleContainer.Visible 2581
OleCtnrs.TOleForm 2613
OleCtnrs.TOleForm.Create 2614
OleCtnrs.TOleForm.Destroy 2614
OleCtnrs.TSizeMode 2621
OleCtnrs.TSizeMode enumeration 2621
OleCtnrs.TVerbInfo 2621
OleCtnrs.TVerbInfo record 2621
OleCtrls 2629
OleCtrls namespace 2629
    Classes 2631
    Functions 2699
    Structs, Records, Enums 2701
    Types 2704
OleCtrls.EmptyParam 2699
OleCtrls.EmptyParam function 2699
OleCtrls.EOleCtrlError 2631
OleCtrls.FontToOleFont 2699
OleCtrls.FontToOleFont function 2699
OleCtrls.OleFontToFont 2700
OleCtrls.OleFontToFont function 2700
OleCtrls.PControlData 2704
OleCtrls.PControlData type 2704

OleCtrls.PControlData2 2704
OleCtrls.PControlData2 type 2704
OleCtrls.PEnumValueList 2705
OleCtrls.PEnumValueList type 2705
OleCtrls.TControlData 2701
OleCtrls.TControlData record 2701
OleCtrls.TControlData2 2702
OleCtrls.TControlData2 record 2702
OleCtrls.TEnumPropDesc 2631
OleCtrls.TEnumPropDesc.Create 2632
OleCtrls.TEnumPropDesc.Destroy 2633
OleCtrls.TEnumPropDesc.GetStrings 2633
OleCtrls.TEnumPropDesc.StringToValue 2634
OleCtrls.TEnumPropDesc.ValueToString 2634
OleCtrls.TEnumValue 2702
OleCtrls.TEnumValue record 2702
OleCtrls.TEnumValueList 2705
OleCtrls.TEnumValueList type 2705
OleCtrls.TEventDispatch 2634
OleCtrls.TEventDispatch._AddRef 2636
OleCtrls.TEventDispatch._Release 2636
OleCtrls.TEventDispatch.Control 2638
OleCtrls.TEventDispatch.Create 2635
OleCtrls.TEventDispatch.GetIDsOfNames 2636
OleCtrls.TEventDispatch.GetTypeInfo 2636
OleCtrls.TEventDispatch.GetTypeInfoCount 2637
OleCtrls.TEventDispatch.Invoke 2637
OleCtrls.TEventDispatch.QueryInterface 2638
OleCtrls.TOleControl 2638
OleCtrls.TOleControl._AddRef 2646
OleCtrls.TOleControl._Release 2647
OleCtrls.TOleControl._SetColorProp 2647
OleCtrls.TOleControl._SetTColorProp 2648
OleCtrls.TOleControl._SetTFontProp 2648
OleCtrls.TOleControl._SetTOleEnumProp 2649
OleCtrls.TOleControl._SetTPictureProp 2649
OleCtrls.TOleControl.Anchors 2695
OleCtrls.TOleControl.BrowseProperties 2649
OleCtrls.TOleControl.CanInPlaceActivate 2650
OleCtrls.TOleControl.ContextSensitiveHelp 2650
OleCtrls.TOleControl.ControlData 2696
OleCtrls.TOleControl.Create 2643
OleCtrls.TOleControl.CreateWnd 2651
OleCtrls.TOleControl.DeactivateAndUndo 2651
OleCtrls.TOleControl.DefaultDispatch 2696
OleCtrls.TOleControl.DefaultHandler 2651
OleCtrls.TOleControl.DefineProperties 2652
OleCtrls.TOleControl.Destroy 2643
OleCtrls.TOleControl.DestroyWindowHandle 2652
OleCtrls.TOleControl.DiscardUndoState 2653
OleCtrls.TOleControl.DoObjectVerb 2653
OleCtrls.TOleControl.EnableModeless 2653
OleCtrls.TOleControl.GetBorder 2654
OleCtrls.TOleControl.GetByteProp 2654
OleCtrls.TOleControl.GetColorProp 2655
OleCtrls.TOleControl.GetCompProp 2655
OleCtrls.TOleControl.GetContainer 2655
OleCtrls.TOleControl.GetCurrencyProp 2656
OleCtrls.TOleControl.GetDoubleProp 2656
OleCtrls.TOleControl.GetEnumPropDesc 2656
OleCtrls.TOleControl.GetExtendedControl 2657
OleCtrls.TOleControl.GetHelpContext 2657
OleCtrls.TOleControl.GetIDispatchProp 2657
OleCtrls.TOleControl.GetIDsOfNames 2658
OleCtrls.TOleControl.GetIntegerProp 2658
OleCtrls.TOleControl.GetIUnknownProp 2658
OleCtrls.TOleControl.GetMoniker 2659
OleCtrls.TOleControl.GetObjectVerbs 2659
OleCtrls.TOleControl.GetOleBoolProp 2659
OleCtrls.TOleControl.GetOleDateProp 2660
OleCtrls.TOleControl.GetOleEnumProp 2660
OleCtrls.TOleControl.GetOleVariantProp 2660
OleCtrls.TOleControl.GetPropDisplayString 2661
OleCtrls.TOleControl.GetPropDisplayStrings 2661
OleCtrls.TOleControl.GetProperty 2661
OleCtrls.TOleControl.GetShortIntProp 2662
OleCtrls.TOleControl.GetSingleProp 2662
OleCtrls.TOleControl.GetSmallintProp 2662

OleCtrls.TOleControl.GetStringProp 2663
OleCtrls.TOleControl.GetTColorProp 2663
OleCtrls.TOleControl.GetTDateTimeProp 2664
OleCtrls.TOleControl.GetTFontProp 2664
OleCtrls.TOleControl.GetTOleEnumProp 2664
OleCtrls.TOleControl.GetTPictureProp 2665
OleCtrls.TOleControl.GetTypeInfo 2665
OleCtrls.TOleControl.GetTypeInfoCount 2665
OleCtrls.TOleControl.GetVariantProp 2666
OleCtrls.TOleControl.GetWideStringProp 2666
OleCtrls.TOleControl.GetWindowContext 2667
OleCtrls.TOleControl.GetWordBoolProp 2667
OleCtrls.TOleControl.GetWordProp 2667
OleCtrls.TOleControl.InitControlData 2668
OleCtrls.TOleControl.InitControlInterface 2668
OleCtrls.TOleControl.InsertMenus 2668
OleCtrls.TOleControl.Invoke 2668
OleCtrls.TOleControl.InvokeEvent 2669
OleCtrls.TOleControl.InvokeMethod 2669
OleCtrls.TOleControl.IsCustomProperty 2669
OleCtrls.TOleControl.IsPropPageProperty 2669
OleCtrls.TOleControl.LockInPlaceActive 2670
OleCtrls.TOleControl.OleControlSite_TranslateAccelerator 2670
OleCtrls.TOleControl.OleInPlaceFrame_GetWindow 2670
OleCtrls.TOleControl.OleInPlaceFrame_TranslateAccelerator 2670
OleCtrls.TOleControl.OleInPlaceSite_GetWindow 2671
OleCtrls.TOleControl.OleObject 2697
OleCtrls.TOleControl.OnChanged 2671
OleCtrls.TOleControl.OnControlInfoChanged 2671
OleCtrls.TOleControl.OnFocus 2671
OleCtrls.TOleControl.OnInPlaceActivate 2672
OleCtrls.TOleControl.OnInPlaceDeactivate 2672
OleCtrls.TOleControl.OnPosRectChange 2672
OleCtrls.TOleControl.OnRequestEdit 2672
OleCtrls.TOleControl.OnShowWindow 2672
OleCtrls.TOleControl.OnUIActivate 2673
OleCtrls.TOleControl.OnUIDeactivate 2673
OleCtrls.TOleControl.PaletteChanged 2673
OleCtrls.TOleControl.PerPropBrowsing 2697
OleCtrls.TOleControl.PictureChanged 2674
OleCtrls.TOleControl.PostMessageFilter 2674
OleCtrls.TOleControl.PreMessageFilter 2674
OleCtrls.TOleControl.QueryInterface 2675
OleCtrls.TOleControl.QueryService 2675
OleCtrls.TOleControl.RemoveMenus 2675
OleCtrls.TOleControl.RequestBorderSpace 2675
OleCtrls.TOleControl.RequestNewObjectLayout 2676
OleCtrls.TOleControl.SaveObject 2676
OleCtrls.TOleControl.Scroll 2676
OleCtrls.TOleControl.ServiceQuery 2698
OleCtrls.TOleControl.SetActiveObject 2677
OleCtrls.TOleControl.SetBorderSpace 2677
OleCtrls.TOleControl.SetBounds 2677
OleCtrls.TOleControl.SetByteProp 2677
OleCtrls.TOleControl.SetColorProp 2678
OleCtrls.TOleControl.SetCompProp 2678
OleCtrls.TOleControl.SetCurrencyProp 2679
OleCtrls.TOleControl.SetDoubleProp 2679
OleCtrls.TOleControl.SetIDispatchProp 2679
OleCtrls.TOleControl.SetIntegerProp 2680
OleCtrls.TOleControl.SetIUnknownProp 2680
OleCtrls.TOleControl.SetMenu 2680
OleCtrls.TOleControl.SetName 2681
OleCtrls.TOleControl.SetOleBoolProp 2681
OleCtrls.TOleControl.SetOleDateProp 2681
OleCtrls.TOleControl.SetOleEnumProp 2682
OleCtrls.TOleControl.SetOleVariantProp 2682
OleCtrls.TOleControl.SetParent 2682
OleCtrls.TOleControl.SetPropDisplayString 2683
OleCtrls.TOleControl.SetProperty 2683
OleCtrls.TOleControl.SetShortIntProp 2684
OleCtrls.TOleControl.SetSingleProp 2684
OleCtrls.TOleControl.SetSmallintProp 2684
OleCtrls.TOleControl.SetStatusText 2685
OleCtrls.TOleControl.SetStringProp 2685
OleCtrls.TOleControl.SetTColorProp 2685
OleCtrls.TOleControl.SetTDateTimeProp 2686
OleCtrls.TOleControl.SetTFontProp 2686

OleCtrls.TOleControl.SetTOleEnumProp 2686
OleCtrls.TOleControl.SetTPictureProp 2687
OleCtrls.TOleControl.SetVariantProp 2687
OleCtrls.TOleControl.SetWideStringProp 2688
OleCtrls.TOleControl.SetWordBoolProp 2688
OleCtrls.TOleControl.SetWordProp 2688
OleCtrls.TOleControl.ShowAboutBox 2689
OleCtrls.TOleControl.ShowObject 2689
OleCtrls.TOleControl.ShowPropertyFrame 2689
OleCtrls.TOleControl.StandardEvent 2690
OleCtrls.TOleControl.SuppressException 2690
OleCtrls.TOleControl.TabStop 2697
OleCtrls.TOleControl.TransformCoords 2690
OleCtrls.TOleControl.WndProc 2691
OleCtrls.TOleEnum 2705
OleCtrls.TOleEnum type 2705
OleCtrls.TServiceQuery 2706
OleCtrls.TServiceQuery type 2706
OleInPlaceFrame_GetWindow
TOleControl class 2670
OleInPlaceFrame_TranslateAccelerator
TOleControl class 2670
OleInPlaceSite_GetWindow
TOleControl class 2671
OleObject
TAutoObject class 2511
TOleContainer class 2569
TOleControl class 2697
OleObjectInterface
TOleContainer class 2569
OleServer 2707
OleServer namespace 2707
Classes 2708
Structs, Records, Enums 2725
Types 2727
OleServer.PServerData 2727
OleServer.PServerData type 2727
OleServer.TConnectKind 2725
OleServer.TConnectKind enumeration 2725
OleServer.TOleServer 2708
OleServer.TOleServer._AddRef 2711
OleServer.TOleServer._Release 2712
OleServer.TOleServer.AutoConnect 2717
OleServer.TOleServer.Connect 2712
OleServer.TOleServer.ConnectEvents 2713
OleServer.TOleServer.ConnectKind 2718
OleServer.TOleServer.Create 2709
OleServer.TOleServer.Destroy 2710
OleServer.TOleServer.Disconnect 2713
OleServer.TOleServer.DisconnectEvents 2713
OleServer.TOleServer.EventDispatch 2718
OleServer.TOleServer.GetAutoConnect 2714
OleServer.TOleServer.GetConnectKind 2714
OleServer.TOleServer.GetServer 2714
OleServer.TOleServer.InitServerData 2715
OleServer.TOleServer.InvokeEvent 2715
OleServer.TOleServer.Loaded 2716
OleServer.TOleServer.QueryInterface 2716
OleServer.TOleServer.RemoteMachineName 2719
OleServer.TOleServer.ServerData 2719
OleServer.TOleServer.SetAutoConnect 2716
OleServer.TOleServer.SetConnectKind 2717
OleServer.TServerData 2725
OleServer.TServerData record 2725
OleServer.TServerEventDispatch 2720
OleServer.TServerEventDispatch._AddRef 2721
OleServer.TServerEventDispatch._Release 2721
OleServer.TServerEventDispatch.Create 2720
OleServer.TServerEventDispatch.GetIDsOfNames 2721
OleServer.TServerEventDispatch.GetTypeInfo 2722
OleServer.TServerEventDispatch.GetTypeInfoCount 2722
OleServer.TServerEventDispatch.Invoke 2723
OleServer.TServerEventDispatch.QueryInterface 2723
OleServer.TServerEventDispatch.Server 2724
OleServer.TServerEventDispatch.ServerDisconnect 2723
OleServer.TVariantArray 2727
OleServer.TVariantArray type 2727
OnActivate

TMtsAutoObject class 2393
TMtsDataModule class 2412
TOleContainer class 2583
OnAdvancedDrawItem
TMenuItem class 1416
OnAfterExecute
TRIO class 4065
OnAfterGetContent
TBaseInetXPageProducer class 2273
TInetXPageProducer class 2293
OnAfterGetXMLData
TBaseInetXPageProducer class 2274
TInetXPageProducer class 2293
OnAttach
TIBCustomService class 433
OnBeforeExecute
TRIO class 4066
OnBeforeGetContent
TBaseInetXPageProducer class 2274
TInetXPageProducer class 2294
OnBeforeGetXMLData
TBaseInetXPageProducer class 2275
TInetXPageProducer class 2294
OnCalcFields
TIBCustomDataSet class 101
TIBDataSet class 129
OnChange
TChangeLink class 650
TCustomImageList class 684
TMainMenu class 1331
TMaskEdit class 1142
TMenu class 1352
TPopupMenu class 1443
TQuickAccessToolbar class 3475
TRibbonComboBox class 3648
TRibbonSpinEdit class 3866
TWebDisplay class 2098
TXMLData class 2161
OnChanged
TOleControl class 2671
OnClick
TMaskEdit class 1142
TMediaPlayer class 2366
TMenuItem class 1417
TOffice2007Button class 3467
TOutline class 2906
TRibbonComboBox class 3649
TRibbonSpinEdit class 3867
OnCollapse
TCustomOutline class 2873
TOutline class 2908
OnCompare
TListControlItems class 1054
OnContextPopup
TMediaPlayer class 2367
TOutline class 2908
TRibbonComboBox class 3651
OnControlInfoChanged
TOleControl class 2671
OnDatabaseFree
TIBBase class 176
OnDataRequest
TBaseProvider class 3091
TCustomProvider class 3124
TDataSetProvider class 3173
OnDblClick
TMaskEdit class 1144
TOutline class 2909
TRibbonComboBox class 3651
TRibbonSpinEdit class 3868
OnDeactivate
TMtsAutoObject class 2393
TMtsDataModule class 2412
TOleContainer class 2583
OnDeleteError
TIBCustomDataSet class 102
TIBDataSet class 129
OnDialectDowngradeWarning

TIBDataBase class 200
OnDownClick
TRibbonSpinButton class 3820
OnDragDrop
TMaskEdit class 1144
TOleContainer class 2584
TOutline class 2909
TRibbonComboBox class 3652
TRibbonSpinButton class 3820
TRibbonSpinEdit class 3869
OnDragOver
TMaskEdit class 1147
TOleContainer class 2587
TOutline class 2912
TRibbonComboBox class 3655
TRibbonSpinButton class 3823
TRibbonSpinEdit class 3872
OnDrawItem
TCustomOutline class 2873
TMenuItem class 1422
TOutline class 2916
OnEditError
TIBCustomDataSet class 102
TIBDataSet class 129
OnEndDock
TMaskEdit class 1150
TOutline class 2916
TRibbonComboBox class 3658
TRibbonSpinButton class 3827
OnEndDrag
TMaskEdit class 1151
TOleContainer class 2590
TOutline class 2917
TRibbonComboBox class 3658
TRibbonSpinButton class 3827
TRibbonSpinEdit class 3875
OnEnter
TMaskEdit class 1152
TMediaPlayer class 2368
TOleContainer class 2591
TOutline class 2918
TRibbonComboBox class 3659
TRibbonSpinButton class 3828
TRibbonSpinEdit class 3876
OneRelationAtATime enumeration member 480
OnError
TIBEvents class 265
TIBSetup class 321
OnEventAlert
TIBEvents class 265
OnExit
TMaskEdit class 1154
TMediaPlayer class 2370
TOleContainer class 2593
TOutline class 2920
TRibbonComboBox class 3662
TRibbonSpinButton class 3831
TRibbonSpinEdit class 3878
OnExpand
TCustomOutline class 2873
TOutline class 2924
OnFilterRecord
TIBDataSet class 129
TIBQuery class 390
TIBStoredProc class 593
TIBTable class 620
OnFocus
TOleControl class 2671
OnGetAttachment
TRIO class 4067
OnGetData
TBaseProvider class 3091
TDataSetProvider class 3174
OnGetDataSetProperties
TDataSetProvider class 3174
OnGetItem
TCustomStaticListAction class 1033
TCustomVirtualListAction class 1037

TStaticListAction class 1061
TVirtualListAction class 1072
OnGetItemCount
TCustomListAction class 1027
TVirtualListAction class 1072
OnGetParams
TDataPacketWriter class 3144
OnGetTableName
TDataSetProvider class 3175
OnGetUsername
TDispatchConnection class 1304
OnHelpButtonClick
TCustomRibbon class 3400
TRibbon class 3498
OnHint
TStaticListAction class 1061
TVirtualListAction class 1073
OnIdleTimer
TIBDataBase class 200
TIBTransaction class 219
OnInPlaceActivate
TOleControl class 2672
OnInPlaceDeactivate
TOleControl class 2672
OnItemSelected
TCustomListAction class 1027
TStaticListAction class 1062
TVirtualListAction class 1073
OnKeyDown
TMaskEdit class 1158
TOleContainer class 2597
TOutline class 2924
TRibbonComboBox class 3666
TRibbonSpinEdit class 3882
OnKeyPress
TMaskEdit class 1162
TOleContainer class 2601
TOutline class 2928
TRibbonComboBox class 3669
TRibbonSpinEdit class 3886
OnKeyUp
TMaskEdit class 1163
TOleContainer class 2602
TOutline class 2929
TRibbonComboBox class 3670
TRibbonSpinEdit class 3887
OnLastRelease
TAutomation class 2507
OnLeftTabScrollButtonClick
TCustomRibbon class 3400
OnLogin
TDispatchConnection class 1304
TIBCustomService class 433
TIBDataBase class 201
OnMeasureItem
TMenuItem class 1423
OnMemberDataNotReceived
TSOAPDomConv class 2782
OnMoreCommands
TCustomRibbonQuickAccessToolbar class 3452
TRibbonQuickAccessToolbar class 3548
OnMouseActivate
TMaskEdit class 1165
TOleContainer class 2604
TOutline class 2931
TRibbonComboBox class 3672
OnMouseDown
TMaskEdit class 1166
TOleContainer class 2605
TOutline class 2932
TRibbonComboBox class 3673
TRibbonSpinEdit class 3889
OnMouseEnter
TMaskEdit class 1168
TMediaPlayer class 2374
TOutline class 2934
TRibbonComboBox class 3675
OnMouseLeave

TMaskEdit class 1168
TMediaPlayer class 2374
TOutline class 2934
TRibbonComboBox class 3676
OnMouseMove
TMaskEdit class 1168
TOleContainer class 2607
TOutline class 2935
TRibbonComboBox class 3676
TRibbonSpinEdit class 3891
OnMouseUp
TMaskEdit class 1171
TOleContainer class 2609
TOutline class 2937
TRibbonComboBox class 3678
TRibbonSpinEdit class 3893
OnNewRecord
TIBCustomDataSet class 102
TIBDataSet class 129
OnNotify
TMediaPlayer class 2375
OnObjectMove
TOleContainer class 2611
OnPopup
TPopupMenu class 1444
OnPosRectChange
TOleControl class 2672
OnPostClick
TMediaPlayer class 2375
OnPostError
TIBCustomDataSet class 102
TIBDataSet class 130
OnRecentItemClick
TCustomRibbon class 3401
TRibbon class 3498
OnRequestEdit
TOleControl class 2672
OnResize
TOleContainer class 2612
OnRightTabScrollButtonClick
TCustomRibbon class 3401
OnSendAttachment
TRIO class 4067
OnShowWindow
TOleControl class 2672
OnSQL
TIBCustomSQLMonitor class 564
TIBSQLMonitor class 566
OnSQLChanging
TIBSQL class 535
OnStartDock
TMaskEdit class 1173
TOutline class 2939
TRibbonComboBox class 3680
TRibbonSpinButton class 3835
OnStartDrag
TMaskEdit class 1173
TOleContainer class 2612
TOutline class 2940
TRibbonComboBox class 3681
TRibbonSpinButton class 3835
TRibbonSpinEdit class 3896
OnStatusChange
TIBSetup class 322
OnTabChange
TCustomRibbon class 3401
TRibbon class 3498
OnTimer
TIBTimer class 206
OnTransactionFree
TIBBase class 176
OnUIActivate
TOleControl class 2673
OnUIDeactivate
TOleControl class 2673
OnUnhandledNode
TSOAPDomConv class 2782
OnUpClick

TRibbonSpinButton class 3836
OnUpdate
TPanningWindow class 704
TStaticListAction class 1062
OnUpdateData
TBaseProvider class 3092
TDataSetProvider class 3175
OnUpdateError
TBaseProvider class 3092
TDataSetProvider class 3176
TIBCustomDataSet class 110
OnUpdateRecord
TIBCustomDataSet class 111
OnValidate
TCustomProvider class 3125
TDataSetProvider class 3176
OnWarning
TIBSetup class 322
ooDrawFocusRect enumeration member 2959
ooDrawTreeRoot enumeration member 2959
ooStretchBitmaps enumeration member 2959
OPConvert 2729
OPConvert namespace 2729
Interfaces 2730
Structs, Records, Enums 2740
Types 2743
OPConvert.IOPConvert 2730
OPConvert.IOPConvert.Attachments 2738
OPConvert.IOPConvert.Encoding 2738
OPConvert.IOPConvert.GetAttachments 2732
OPConvert.IOPConvert.GetEncoding 2732
OPConvert.IOPConvert.GetOptions 2732
OPConvert.IOPConvert.GetTempDir 2732
OPConvert.IOPConvert.InvContextToMsg 2733
OPConvert.IOPConvert.MakeFault 2733
OPConvert.IOPConvert.MakeResponse 2734
OPConvert.IOPConvert.MsgToInvContext 2735
OPConvert.IOPConvert.Options 2739
OPConvert.IOPConvert.ProcessResponse 2736
OPConvert.IOPConvert.SetAttachments 2736
OPConvert.IOPConvert.SetEncoding 2737
OPConvert.IOPConvert.SetOptions 2737
OPConvert.IOPConvert.SetTempDir 2737
OPConvert.IOPConvert.TempDir 2739
OPConvert.TSOAPConvertOption 2740
OPConvert.TSOAPConvertOption enumeration 2740
OPConvert.TSOAPConvertOptions 2743
OPConvert.TSOAPConvertOptions type 2743
Open
TIBSQL class 532
TMediaPlayer class 2319
OpenBlob
TIBBlobStream class 46
Opened
TUpdateTree class 3211
OpenKey
TRegistry class 3270
OpenKeyReadOnly
TRegistry class 3272
OptionItems
TRibbonApplicationMenuBar class 3956
Options
IOPConvert interface 2739
TBaseProvider class 3087
TCustomOutline class 2866
TDataPacketWriter class 3144
TDataSetProvider class 3161
TIBBackupService class 411
TIBRestoreService class 442
TIBServerProperties class 457
TIBStatisticalService class 460
TIBValidationService class 465
TOutline class 2890
TSOAPDomConv class 2781
OPToSOAPDomConv 2745
OPToSOAPDomConv namespace 2745
Classes 2747
Constants 2790

Functions 2783
Structs, Records, Enums 2784
Types 2786
Variables 2789
OPToSOAPDomConv.ConvNodeState 2784
OPToSOAPDomConv.ConvNodeState enumeration 2784
OPToSOAPDomConv.DefArrayElemName 2789
OPToSOAPDomConv.DefArrayElemName variable 2789
OPToSOAPDomConv.ESOAPDomConvertError 2747
OPToSOAPDomConv.GetOrdPropEx 2783
OPToSOAPDomConv.GetOrdPropEx function 2783
OPToSOAPDomConv.SVarArrayType 2790
OPToSOAPDomConv.SVarArrayType constant 2790
OPToSOAPDomConv.TMemberDataNotReceivedEvent 2786
OPToSOAPDomConv.TMemberDataNotReceivedEvent type 2786
OPToSOAPDomConv.TMultiRefNodeElem 2784
OPToSOAPDomConv.TMultiRefNodeElem record 2784
OPToSOAPDomConv.TMultiRefNodeMap 2787
OPToSOAPDomConv.TMultiRefNodeMap type 2787
OPToSOAPDomConv.TMultiRefNodeMapElem 2785
OPToSOAPDomConv.TMultiRefNodeMapElem record 2785
OPToSOAPDomConv.TMultiRefNodes 2787
OPToSOAPDomConv.TMultiRefNodes type 2787
OPToSOAPDomConv.TOPToSoapDomConvert 2747
OPToSOAPDomConv.TOPToSoapDomConvert.Attachments 2756
OPToSOAPDomConv.TOPToSoapDomConvert.Create 2751
OPToSOAPDomConv.TOPToSoapDomConvert.Destroy 2751
OPToSOAPDomConv.TOPToSoapDomConvert.Encoding 2756
OPToSOAPDomConv.TOPToSoapDomConvert.InvContextToMsg 2752
OPToSOAPDomConv.TOPToSoapDomConvert.MakeFault 2752
OPToSOAPDomConv.TOPToSoapDomConvert.MakeResponse 2753
OPToSOAPDomConv.TOPToSoapDomConvert.MsgToInvContext 2754
OPToSOAPDomConv.TOPToSoapDomConvert.NewXMLDocument 2754
OPToSOAPDomConv.TOPToSoapDomConvert.ProcessResponse 2755
OPToSOAPDomConv.TOPToSoapDomConvert.TempDir 2757
OPToSOAPDomConv.TOPToSoapDomConvert.WSDLView 2758
OPToSOAPDomConv.TSOAPArrayDesc 2787
OPToSOAPDomConv.TSOAPArrayDesc type 2787
OPToSOAPDomConv.TSOAPArrayElemDesc 2785
OPToSOAPDomConv.TSOAPArrayElemDesc record 2785
OPToSOAPDomConv.TSOAPDomConv 2758
OPToSOAPDomConv.TSOAPDomConv.AddAttachment 2763
OPToSOAPDomConv.TSOAPDomConv.AddMultiRefNode 2763
OPToSOAPDomConv.TSOAPDomConv.AddNamespaceDecl 2763
OPToSOAPDomConv.TSOAPDomConv.AddObjectAsWriting 2763
OPToSOAPDomConv.TSOAPDomConv.CheckEncodingStyle 2764
OPToSOAPDomConv.TSOAPDomConv.ChildNodesAreNull 2764
OPToSOAPDomConv.TSOAPDomConv.ConvertByteArrayToSoap 2764
OPToSOAPDomConv.TSOAPDomConv.ConvertEnumToSoap 2764
OPToSOAPDomConv.TSOAPDomConv.ConvertNativeArrayToSoap 2765
OPToSOAPDomConv.TSOAPDomConv.ConvertNativeDataToSoap 2765
OPToSOAPDomConv.TSOAPDomConv.ConvertObjectToSOAP 2765
OPToSOAPDomConv.TSOAPDomConv.ConvertSoapToEnum 2766
OPToSOAPDomConv.TSOAPDomConv.ConvertSoapToNativeArray 2766
OPToSOAPDomConv.TSOAPDomConv.ConvertSoapToNativeArrayElem 2766
OPToSOAPDomConv.TSOAPDomConv.ConvertSoapToNativeData 2766

OPToSOAPDomConv.TSOAPDomConv.ConvertSOAPToObject 2767
OPToSOAPDomConv.TSOAPDomConv.ConvertSoapToVariant 2767
OPToSOAPDomConv.TSOAPDomConv.ConvertVariantToSoap 2768
OPToSOAPDomConv.TSOAPDomConv.Create 2760
OPToSOAPDomConv.TSOAPDomConv.CreateMultiRefNode 2768
OPToSOAPDomConv.TSOAPDomConv.CreateNULLNode 2768
OPToSOAPDomConv.TSOAPDomConv.CreateObjectNode 2768
OPToSOAPDomConv.TSOAPDomConv.CreateScalarNodeXS 2769
OPToSOAPDomConv.TSOAPDomConv.CreateTypedNode 2769
OPToSOAPDomConv.TSOAPDomConv.Destroy 2761
OPToSOAPDomConv.TSOAPDomConv.FinalizeMultiRefNodes 2769
OPToSOAPDomConv.TSOAPDomConv.FindAttachment 2769
OPToSOAPDomConv.TSOAPDomConv.FindMultiRefNodeByID 2770
OPToSOAPDomConv.TSOAPDomConv.FindMultiRefNodeByInstance 2770
OPToSOAPDomConv.TSOAPDomConv.FindNodeByHREF 2770
OPToSOAPDomConv.TSOAPDomConv.FindPrefixForURI 2770
OPToSOAPDomConv.TSOAPDomConv.GetDataNode 2771
OPToSOAPDomConv.TSOAPDomConv.GetElementType 2771
OPToSOAPDomConv.TSOAPDomConv.GetNewID 2771
OPToSOAPDomConv.TSOAPDomConv.GetNodeAsText 2771
OPToSOAPDomConv.TSOAPDomConv.GetObjectPropAsText 2771
OPToSOAPDomConv.TSOAPDomConv.GetOptions 2772
OPToSOAPDomConv.TSOAPDomConv.GetTypeBySchemaNS 2772
OPToSOAPDomConv.TSOAPDomConv.InitObjectFromSOAP 2772
OPToSOAPDomConv.TSOAPDomConv.IsNodeAVarArray 2773
OPToSOAPDomConv.TSOAPDomConv.IsObjectWriting 2773
OPToSOAPDomConv.TSOAPDomConv.LoadObject 2773
OPToSOAPDomConv.TSOAPDomConv.MakeArrayNode 2774
OPToSOAPDomConv.TSOAPDomConv.MultiRefObject 2774
OPToSOAPDomConv.TSOAPDomConv.NodeIsNULL 2774
OPToSOAPDomConv.TSOAPDomConv.ObjectMemberNoShow 2775
OPToSOAPDomConv.TSOAPDomConv.ObjInstanceToSOAP 2775
OPToSOAPDomConv.TSOAPDomConv.OnMemberDataNotReceived 2782
OPToSOAPDomConv.TSOAPDomConv.OnUnhandledNode 2782
OPToSOAPDomConv.TSOAPDomConv.Options 2781
OPToSOAPDomConv.TSOAPDomConv.ReadHeader 2775
OPToSOAPDomConv.TSOAPDomConv.ReadRectDynArray 2776
OPToSOAPDomConv.TSOAPDomConv.ReadRectDynArrayElem 2776
OPToSOAPDomConv.TSOAPDomConv.ReadRow 2776
OPToSOAPDomConv.TSOAPDomConv.ReadVarArrayDim 2776
OPToSOAPDomConv.TSOAPDomConv.ReadVariant 2777
OPToSOAPDomConv.TSOAPDomConv.RemoveObjectAsWriting 2777
OPToSOAPDomConv.TSOAPDomConv.ResetMultiRef 2777
OPToSOAPDomConv.TSOAPDomConv.SerializationOptions 2778
OPToSOAPDomConv.TSOAPDomConv.SetNodeType 2778
OPToSOAPDomConv.TSOAPDomConv.SetObjectPropFromText 2778
OPToSOAPDomConv.TSOAPDomConv.SetOptions 2779
OPToSOAPDomConv.TSOAPDomConv.UnhandledNode 2779
OPToSOAPDomConv.TSOAPDomConv.WriteHeader 2779
OPToSOAPDomConv.TSOAPDomConv.WriteNonRectDynArray 2779
OPToSOAPDomConv.TSOAPDomConv.WriteNonRectDynArrayElem 2780
OPToSOAPDomConv.TSOAPDomConv.WriteRectDynArray 2780

OPToSOAPDomConv.TSOAPDomConv.WriteRectDynArrayElem 2780
OPToSOAPDomConv.TSOAPDomConv.WriteVarArray 2780
OPToSOAPDomConv.TSOAPDomConv.WriteVarArrayAsB64 2781
OPToSOAPDomConv.TSOAPDomConv.WriteVariant 2781
OPToSOAPDomConv.TUnhandledNodeEvent 2788
OPToSOAPDomConv.TUnhandledNodeEvent type 2788
OPToSOAPDomConv.TXMLNodeArray 2788
OPToSOAPDomConv.TXMLNodeArray type 2788
Orb
IORB interface 2808
OrbPas 2791
OrbPas namespace 2791
Functions 2821
Interfaces 2793
Structs, Records, Enums 2826
Types 2828
OrbPas.AnyToVariant 2821
OrbPas.AnyToVariant function 2821
OrbPas.BindStub 2821
OrbPas.BindStub function 2821
OrbPas.CorbaAnyType 2822
OrbPas.CorbaAnyType function 2822
OrbPas.CorbaBoolean 2828
OrbPas.CorbaBoolean type 2828
OrbPas.CorbaDuplicateAny 2822
OrbPas.CorbaDuplicateAny function 2822
OrbPas.CorbaReleaseAny 2822
OrbPas.CorbaReleaseAny function 2822
OrbPas.CorbaStringFree 2823
OrbPas.CorbaStringFree function 2823
OrbPas.CorbaULong 2828
OrbPas.CorbaULong type 2828
OrbPas.CorbaWStringFree 2823
OrbPas.CorbaWStringFree function 2823
OrbPas.CreateSkeleton 2823
OrbPas.CreateSkeleton function 2823
OrbPas.CreateUserException 2824
OrbPas.CreateUserException function 2824
OrbPas.IBOA 2793
OrbPas.IBOA.Deactivate 2794
OrbPas.IBOA.GetPrincipal 2794
OrbPas.IBOA.GetPrincipalLength 2794
OrbPas.IBOA.ImplIsReady 2794
OrbPas.IBOA.ObjIsReady 2795
OrbPas.ICorbaObj 2795
OrbPas.ICorbaObj.CorbaObject 2796
OrbPas.ICorbaObj.IsLocal 2796
OrbPas.IMarshalInBuffer 2796
OrbPas.IMarshalInBuffer.Buffer 2797
OrbPas.IMarshalInBuffer.GetAny 2798
OrbPas.IMarshalInBuffer.GetChar 2798
OrbPas.IMarshalInBuffer.GetDouble 2798
OrbPas.IMarshalInBuffer.GetFloat 2798
OrbPas.IMarshalInBuffer.GetLong 2798
OrbPas.IMarshalInBuffer.GetObject 2799
OrbPas.IMarshalInBuffer.GetShort 2799
OrbPas.IMarshalInBuffer.GetText 2799
OrbPas.IMarshalInBuffer.GetUnsignedChar 2799
OrbPas.IMarshalInBuffer.GetUnsignedLong 2800
OrbPas.IMarshalInBuffer.GetUnsignedShort 2800
OrbPas.IMarshalInBuffer.GetWidechar 2800
OrbPas.IMarshalInBuffer.GetWideText 2800
OrbPas.IMarshalOutBuffer 2800
OrbPas.IMarshalOutBuffer.Buffer 2802
OrbPas.IMarshalOutBuffer.PutAny 2802
OrbPas.IMarshalOutBuffer.PutChar 2802
OrbPas.IMarshalOutBuffer.PutDouble 2802
OrbPas.IMarshalOutBuffer.PutFloat 2802
OrbPas.IMarshalOutBuffer.PutLong 2803
OrbPas.IMarshalOutBuffer.PutObject 2803
OrbPas.IMarshalOutBuffer.PutShort 2803
OrbPas.IMarshalOutBuffer.PutText 2803
OrbPas.IMarshalOutBuffer.PutUnsignedChar 2804
OrbPas.IMarshalOutBuffer.PutUnsignedLong 2804
OrbPas.IMarshalOutBuffer.PutUnsignedShort 2804
OrbPas.IMarshalOutBuffer.PutWidechar 2804

OrbPas.IMarshalOutBuffer.PutWideText 2804
OrbPas.InitORB 2824
OrbPas.InitORB function 2824
OrbPas.IORB 2805
OrbPas.IORB.BOAInit 2806
OrbPas.IORB.CreateAliasTC 2806
OrbPas.IORB.CreateArrayTC 2806
OrbPas.IORB.CreateObjRefTC 2807
OrbPas.IORB.CreateSequenceTC 2807
OrbPas.IORB.CreateStructTC 2807
OrbPas.IORB.CreateTC 2807
OrbPas.IORB.DispatchStruct 2807
OrbPas.IORB.FindRepositoryTC 2808
OrbPas.IORB.MakeAny 2808
OrbPas.IORB.ObjectToString 2808
OrbPas.IORB.Orb 2808
OrbPas.IORB.Shutdown 2809
OrbPas.IORB.StringToObject 2809
OrbPas.ISkeleton 2809
OrbPas.ISkeleton.GetImplementation 2810
OrbPas.ISkeleton.GetReplyBuffer 2810
OrbPas.ISkeletonObject 2810
OrbPas.ISkeletonObject.Execute 2811
OrbPas.ISkeletonObject.GetImplementation 2811
OrbPas.ISkeletonObject.GetSkeleton 2811
OrbPas.IStub 2812
OrbPas.IStub.CreateRequest 2813
OrbPas.IStub.Dispatch 2813
OrbPas.IStub.GetInterface 2813
OrbPas.IStub.Hash 2813
OrbPas.IStub.Invoke 2814
OrbPas.IStub.IsA 2814
OrbPas.IStub.NonExistent 2814
OrbPas.IStub.RepositoryID 2814
OrbPas.IStub.SetPrincipal 2815
OrbPas.IStubObject 2815
OrbPas.IStubObject.GetStub 2816
OrbPas.ITypeCode 2816
OrbPas.ITypeCode.ContentType 2817
OrbPas.ITypeCode.DefaultIndex 2817
OrbPas.ITypeCode.DiscriminatorType 2817
OrbPas.ITypeCode.Equal 2818
OrbPas.ITypeCode.Id 2818
OrbPas.ITypeCode.Kind 2818
OrbPas.ITypeCode.Length 2818
OrbPas.ITypeCode.MemberCount 2819
OrbPas.ITypeCode.MemberLabel 2819
OrbPas.ITypeCode.MemberName 2819
OrbPas.ITypeCode.MemberType 2819
OrbPas.ITypeCode.Name 2819
OrbPas.ITypeCode.Parameter 2820
OrbPas.ITypeCode.ParameterCount 2820
OrbPas.PCorbaAny 2829
OrbPas.PCorbaAny type 2829
OrbPas.PCorbaInterfaceDef 2829
OrbPas.PCorbaInterfaceDef type 2829
OrbPas.PCorbaObject 2829
OrbPas.PCorbaObject type 2829
OrbPas.PCorbaOrb 2830
OrbPas.PCorbaOrb type 2830
OrbPas.PExceptionDescription 2830
OrbPas.PExceptionDescription type 2830
OrbPas.PMarshalInbuffer 2830
OrbPas.PMarshalInbuffer type 2830
OrbPas.PMarshalOutbuffer 2831
OrbPas.PMarshalOutbuffer type 2831
OrbPas.PUserExceptionProxy 2831
OrbPas.PUserExceptionProxy type 2831
OrbPas.RegisterUserException 2824
OrbPas.RegisterUserException function 2824
OrbPas.SequenceToVariant 2825
OrbPas.SequenceToVariant function 2825
OrbPas.TArgv 2831
OrbPas.TArgv type 2831
OrbPas.TCKind 2826
OrbPas.TCKind enumeration 2826
OrbPas.TCopyUserExceptionProc 2832
OrbPas.TCopyUserExceptionProc type 2832

OrbPas.TStructMember 2827
OrbPas.TStructMember record 2827
OrbPas.TStructMembers 2832
OrbPas.TStructMembers type 2832
OrbPas.TThrowUserExceptionProc 2832
OrbPas.TThrowUserExceptionProc type 2832
OrbPas.TUserExceptionFactoryProc 2833
OrbPas.TUserExceptionFactoryProc type 2833
OrbPas.UnRegisterUserException 2825
OrbPas.UnRegisterUserException function 2825
OrbPas.VariantToAny 2825
OrbPas.VariantToAny function 2825
Orientation
    TPrinter class 3056
osEmpty enumeration member 2620
osInPlaceActive enumeration member 2620
osLoaded enumeration member 2620
osOpen enumeration member 2620
osPictureText enumeration member 2959
osPlusMinusPictureText enumeration member 2959
osPlusMinusText enumeration member 2959
osRunning enumeration member 2620
osText enumeration member 2959
osTreePictureText enumeration member 2959
osTreeText enumeration member 2959
osUIActive enumeration member 2620
otOwnerDraw enumeration member 2960
otStandard enumeration member 2960
Outline 2835
    TOutlineNode class 2955
Outline namespace 2835
    Classes 2837
    Structs, Records, Enums 2957
    Types 2962
Outline.EOutlineChange 2962
Outline.EOutlineChange type 2962
Outline.EOutlineError 2837
Outline.OutlineError 2838
Outline.TAttachMode 2957
Outline.TAttachMode enumeration 2957
Outline.TBitmapArray 2962
Outline.TBitmapArray type 2962
Outline.TBitmapArrayRange 2963
Outline.TBitmapArrayRange type 2963
Outline.TChangeRange 2963
Outline.TChangeRange type 2963
Outline.TCustomOutline 2838
Outline.TCustomOutline.Add 2842
Outline.TCustomOutline.AddChild 2843
Outline.TCustomOutline.AddChildObject 2844
Outline.TCustomOutline.AddObject 2844
Outline.TCustomOutline.BadIndex 2845
Outline.TCustomOutline.BeginUpdate 2846
Outline.TCustomOutline.Canvas 2862
Outline.TCustomOutline.Clear 2846
Outline.TCustomOutline.Click 2846
Outline.TCustomOutline.Collapse 2847
Outline.TCustomOutline.Create 2840
Outline.TCustomOutline.DblClick 2847
Outline.TCustomOutline.DefineProperties 2847
Outline.TCustomOutline.Delete 2848
Outline.TCustomOutline.DeleteNode 2848
Outline.TCustomOutline.Destroy 2841
Outline.TCustomOutline.DrawCell 2849
Outline.TCustomOutline.EndUpdate 2849
Outline.TCustomOutline.Expand 2850
Outline.TCustomOutline.FullCollapse 2850
Outline.TCustomOutline.FullExpand 2850
Outline.TCustomOutline.GetDataItem 2851
Outline.TCustomOutline.GetItem 2851
Outline.TCustomOutline.GetNodeDisplayWidth 2852
Outline.TCustomOutline.GetTextItem 2852
Outline.TCustomOutline.GetVisibleNode 2852
Outline.TCustomOutline.Insert 2853
Outline.TCustomOutline.InsertObject 2853
Outline.TCustomOutline.InvalidateCell 2854
Outline.TCustomOutline.ItemCount 2864
Outline.TCustomOutline.ItemHeight 2864

Outline.TCustomOutline.Items 2865
Outline.TCustomOutline.ItemSeparator 2865
Outline.TCustomOutline.KeyDown 2855
Outline.TCustomOutline.KeyPress 2855
Outline.TCustomOutline.Lines 2866
Outline.TCustomOutline.Loaded 2856
Outline.TCustomOutline.LoadFromFile 2856
Outline.TCustomOutline.LoadFromStream 2857
Outline.TCustomOutline.MouseDown 2858
Outline.TCustomOutline.Move 2858
Outline.TCustomOutline.OnCollapse 2873
Outline.TCustomOutline.OnDrawItem 2873
Outline.TCustomOutline.OnExpand 2873
Outline.TCustomOutline.Options 2866
Outline.TCustomOutline.OutlineStyle 2867
Outline.TCustomOutline.PictureClosed 2868
Outline.TCustomOutline.PictureLeaf 2868
Outline.TCustomOutline.PictureMinus 2869
Outline.TCustomOutline.PictureOpen 2869
Outline.TCustomOutline.PicturePlus 2870
Outline.TCustomOutline.Row 2870
Outline.TCustomOutline.SaveToFile 2859
Outline.TCustomOutline.SaveToStream 2859
Outline.TCustomOutline.SelectedItem 2872
Outline.TCustomOutline.SetDisplayWidth 2860
Outline.TCustomOutline.SetGoodIndex 2860
Outline.TCustomOutline.SetLevel 2861
Outline.TCustomOutline.SetUpdateState 2861
Outline.TCustomOutline.Style 2872
Outline.TOutline 2874
Outline.TOutline.Align 2878
Outline.TOutline.BorderStyle 2881
Outline.TOutline.Color 2881
Outline.TOutline.Ctl3D 2882
Outline.TOutline.DoubleBuffered 2883
Outline.TOutline.DragCursor 2885
Outline.TOutline.DragKind 2885
Outline.TOutline.DragMode 2885
Outline.TOutline.Enabled 2887
Outline.TOutline.Font 2887
Outline.TOutline.ItemHeight 2888
Outline.TOutline.ItemSeparator 2889
Outline.TOutline.Lines 2889
Outline.TOutline.OnClick 2906
Outline.TOutline.OnCollapse 2908
Outline.TOutline.OnContextPopup 2908
Outline.TOutline.OnDblClick 2909
Outline.TOutline.OnDragDrop 2909
Outline.TOutline.OnDragOver 2912
Outline.TOutline.OnDrawItem 2916
Outline.TOutline.OnEndDock 2916
Outline.TOutline.OnEndDrag 2917
Outline.TOutline.OnEnter 2918
Outline.TOutline.OnExit 2920
Outline.TOutline.OnExpand 2924
Outline.TOutline.OnKeyDown 2924
Outline.TOutline.OnKeyPress 2928
Outline.TOutline.OnKeyUp 2929
Outline.TOutline.OnMouseActivate 2931
Outline.TOutline.OnMouseDown 2932
Outline.TOutline.OnMouseEnter 2934
Outline.TOutline.OnMouseLeave 2934
Outline.TOutline.OnMouseMove 2935
Outline.TOutline.OnMouseUp 2937
Outline.TOutline.OnStartDock 2939
Outline.TOutline.OnStartDrag 2940
Outline.TOutline.Options 2890
Outline.TOutline.OutlineStyle 2890
Outline.TOutline.ParentColor 2891
Outline.TOutline.ParentCtl3D 2892
Outline.TOutline.ParentDoubleBuffered 2892
Outline.TOutline.ParentFont 2892
Outline.TOutline.ParentShowHint 2893
Outline.TOutline.PictureClosed 2893
Outline.TOutline.PictureLeaf 2894
Outline.TOutline.PictureMinus 2894
Outline.TOutline.PictureOpen 2895
Outline.TOutline.PicturePlus 2895

Outline.TOutline.PopupMenu 2896
Outline.TOutline.ScrollBars 2901
Outline.TOutline.ShowHint 2901
Outline.TOutline.Style 2902
Outline.TOutline.TabOrder 2903
Outline.TOutline.TabStop 2904
Outline.TOutline.Visible 2904
Outline.TOutlineBitmap 2958
Outline.TOutlineBitmap enumeration 2958
Outline.TOutlineBitmaps 2963
Outline.TOutlineBitmaps type 2963
Outline.TOutlineNode 2940
Outline.TOutlineNode.AddNode 2944
Outline.TOutlineNode.ChangeLevelBy 2944
Outline.TOutlineNode.Collapse 2944
Outline.TOutlineNode.Create 2942
Outline.TOutlineNode.Data 2951
Outline.TOutlineNode.Destroy 2943
Outline.TOutlineNode.Expand 2945
Outline.TOutlineNode.ExpandCount 2952
Outline.TOutlineNode.Expanded 2952
Outline.TOutlineNode.FullExpand 2945
Outline.TOutlineNode.FullPath 2952
Outline.TOutlineNode.GetDataItem 2946
Outline.TOutlineNode.GetDisplayWidth 2946
Outline.TOutlineNode.getFirstChild 2946
Outline.TOutlineNode.GetLastChild 2947
Outline.TOutlineNode.GetNextChild 2947
Outline.TOutlineNode.GetNodeAtIndex 2947
Outline.TOutlineNode.GetPrevChild 2948
Outline.TOutlineNode.GetRowOfNode 2948
Outline.TOutlineNode.GetTextItem 2948
Outline.TOutlineNode.GetVisibleNode 2949
Outline.TOutlineNode.HasAsParent 2949
Outline.TOutlineNode.HasItems 2953
Outline.TOutlineNode.Index 2953
Outline.TOutlineNode.InsertNode 2949
Outline.TOutlineNode.InternalRemove 2949
Outline.TOutlineNode.IsVisible 2954
Outline.TOutlineNode.Items 2954
Outline.TOutlineNode.Level 2954
Outline.TOutlineNode.List 2955
Outline.TOutlineNode.MoveTo 2949
Outline.TOutlineNode.Outline 2955
Outline.TOutlineNode.Parent 2955
Outline.TOutlineNode.Remove 2950
Outline.TOutlineNode.Text 2955
Outline.TOutlineNode.TopItem 2956
Outline.TOutlineNode.WriteNode 2950, 2951
Outline.TOutlineNodeCompare 2958
Outline.TOutlineNodeCompare enumeration 2958
Outline.TOutlineOption 2959
Outline.TOutlineOption enumeration 2959
Outline.TOutlineOptions 2964
Outline.TOutlineOptions type 2964
Outline.TOutlineStyle 2959
Outline.TOutlineStyle enumeration 2959
Outline.TOutlineType 2960
Outline.TOutlineType enumeration 2960
OutlineError class 2838
  about OutlineError class 2838
OutlineStyle
  TCustomOutline class 2867
  TOutline class 2890
OutputBuffer
  TIBCustomService class 431
OutputBufferOption
  TIBCustomService class 431
OutputTitles
  TIBOutputDelimitedFile class 511
Overlay
  TCustomImageList class 671
Owner
  TCustomTabData class 3458
  TIBBase class 174
  TPNGPointerList class 3002
OwnerData
  TCustomProvider class 3115

TDataSetProvider class 3162
OwnerDraw
TMainMenu class 1330
TMenu class 1350
TPopupMenu class 1440
Ownership
TSOAPAttachment class 907
OwnsObjects
THeaderList class 829
OwnsSentHeaders
ISOAPHeaders interface 927

P

paCenter enumeration member 1463
PacketDataSet
TBaseProvider class 3088
PacketOptions
TDataPacketWriter class 3144
Page
TCustomRibbonGroup class 3423
TRibbonTabItem class 3559
PageBuffers
TIBRestoreService class 442
PageElements
TBaseInetXPageProducer class 2271
PageHeight
TPrinter class 3057
PageNumber
TPrinter class 3059
PageSize
TIBDatabaseInfo class 252
TIBRestoreService class 443
PageWidth
TPrinter class 3060
PagItems 2965
PagItems namespace 2965
Classes 2966
PagItems.TBasePageItemsProducer 2966
PagItems.TBasePageItemsProducer.Create 2967
PagItems.TBasePageItemsProducer.Destroy 2967
PagItems.TBasePageItemsProducer.GetChildren 2968
PagItems.TBasePageItemsProducer.GetComponentList 2968
PagItems.TBasePageItemsProducer.GetDefaultComponentList 2969
PagItems.TBasePageItemsProducer.SetChildOrder 2969
PagItems.TBasePageItemsProducer.WebPageItems 2969
PagItems.TPageItemsProducer 2970
PagItems.TPageItemsProducer.Create 2971
PagItems.TPageItemsProducer.Destroy 2971
PagItems.TPageItemsProducer.GetTemplateStream 2972
PagItems.TPageItemsProducer.HTMLDoc 2972
PagItems.TPageItemsProducer.HTMLFile 2973
Paint
TCustomRibbon class 3387
TCustomRibbonGroup class 3418
TCustomRibbonPage class 3433
TMediaPlayer class 2322
TOffice2007Button class 3466
TOleContainer class 2546
TPanningWindow class 697
TRibbonApplicationMenuBar class 3948
TRibbonApplicationMenuButton class 3609
TRibbonComboControl class 3693
TRibbonCustomControl class 3705
TRibbonGalleryControlMenuItem class 3724
TRibbonGripItem class 3737
TRibbonGroupCollapsedControl class 3513
TRibbonGroupScrollButton class 3523
TRibbonRecentDocuments class 3972
TTimerSpeedButton class 3910
paLeft enumeration member 1463
PaletteChanged
TOleControl class 2673
PanControl
TPanningWindow class 703
PanInterval
TPanningWindow class 704
PanOptions
TPanningWindow class 704

ParamByName
TIBDataSet class 120
TIBQuery class 384
TIBSQL class 527
TIBStoredProc class 589
ParamCheck
TIBCustomDataSet class 102
TIBDataSet class 130
TIBQuery class 390
TIBSQL class 532
ParamCount
IInetXWebDataSet interface 2216
TIBQuery class 390
TIBStoredProc class 593
Parameter
ITypeCode interface 2820
ParameterCount
ITypeCode interface 2820
ParamName
IQueryField interface 2219
TQueryRadioGroup class 1985
TQuerySelectOptions class 1997
TQueryText class 2008
TQueryTextArea class 2019
TWebDataInput class 2094
Params
IInetXWebDataSet interface 2216
TDataSetProvider class 3162
TIBBatch class 497
TIBCustomDataSet class 103
TIBCustomService class 431
TIBDataBase class 196
TIBDataSet class 130
TIBQuery class 391
TIBSQL class 533
TIBStoredProc class 593
TIBTransaction class 218
Parent
TCaptionAttributes class 1806
TGridAttributes class 1928
TGridRowAttributes class 1933
TMenuItem class 1414
TOutlineNode class 2955
TUpdateTree class 3212
TWebDisplay class 2097
TXMLData class 2160
TXMLDataSet class 2171
ParentBiDiMode
TMainMenu class 1330
TMaskEdit class 1125
TMenu class 1351
TPopupMenu class 1440
TRibbon class 3495
TRibbonGroup class 3505
TRibbonPage class 3545
ParentBiDiModeChanged
TMenu class 1341
ParentColor
TMaskEdit class 1126
TOleContainer class 2570
TOutline class 2891
TRibbonSpinEdit class 3854
ParentConnection
TSharedConnection class 1314
ParentControl
TRibbonGroupDockForm class 3521
ParentCtl3D
TMaskEdit class 1126
TOleContainer class 2570
TOutline class 2892
TRibbonSpinButton class 3810
TRibbonSpinEdit class 3855
ParentDataSetField
TXMLDataSetParent class 2175
ParentDoubleBuffered
TMaskEdit class 1127
TMediaPlayer class 2352
TOutline class 2892

ParentFont
TMaskEdit class 1127
TOutline class 2892
TRibbon class 3495
TRibbonComboBox class 3636
TRibbonSpinEdit class 3855
ParentShowHint
TMaskEdit class 1127
TMediaPlayer class 2352
TOleContainer class 2571
TOutline class 2893
TRibbonSpinButton class 3810
TRibbonSpinEdit class 3855
ParentXMLBroker
TXMLDataParent class 2163
TXMLDataSetParent class 2175
ParentXMLComponent
TQueryButton class 1959
ParentXMLDataSet
TXMLDataSetParent class 2175
paRight enumeration member 1463
ParseMultiPart
TMultipartContentParser class 3346
ParseMultiPartContent
TMultipartContentParser class 3346
ParseMultipartHeaders
TMultipartContentParser class 3346
Partial
TKeyTipList class 3596
Password
TIBSecurityService class 450
PasswordChar
TMaskEdit class 1128
TRibbonComboBox class 3636
Paste
TOleContainer class 2546
PasteSpecialDialog
TOleContainer class 2547
Pause
TMediaPlayer class 2322
PauseOnly
TMediaPlayer class 2323
pcCollation enumeration member 3070
pcCopies enumeration member 3070
pcOrientation enumeration member 3070
pdDown enumeration member 707
pdLeft enumeration member 707
pdRight enumeration member 707
pdUp enumeration member 707
Performance
TJPEGImage class 1008
PerPropBrowsing
TOleControl class 2697
pfAverage enumeration member 3020
pfNone enumeration member 3020
pfPaeth enumeration member 3020
pfSub enumeration member 3020
pfUp enumeration member 3020
PictureChanged
TOleControl class 2674
PictureClosed
TCustomOutline class 2868
TOutline class 2893
PictureLeaf
TCustomOutline class 2868
TOutline class 2894
PictureMinus
TCustomOutline class 2869
TOutline class 2894
PictureOpen
TCustomOutline class 2869
TOutline class 2895
PicturePlus
TCustomOutline class 2870
TOutline class 2895
PixelFormat
TJPEGImage class 1009
Plan

TIBCustomDataSet class 103
TIBSQL class 533
PlatformDefaultStyleActnCtrls 2975
PlatformDefaultStyleActnCtrls namespace 2975
Classes 2976
Variables 2979
PlatformDefaultStyleActnCtrls.PlatformDefaultStyle 2979
PlatformDefaultStyleActnCtrls.PlatformDefaultStyle variable 2979
PlatformDefaultStyleActnCtrls.TPlatformDefaultStyleActionBars 2976
PlatformDefaultStyleActnCtrls.TPlatformDefaultStyleActionBars.GetAddRemoveItemClass 2977
PlatformDefaultStyleActnCtrls.TPlatformDefaultStyleActionBars.GetColorMapClass 2977
PlatformDefaultStyleActnCtrls.TPlatformDefaultStyleActionBars.GetControlClass 2977
PlatformDefaultStyleActnCtrls.TPlatformDefaultStyleActionBars.GetPopupClass 2978
PlatformDefaultStyleActnCtrls.TPlatformDefaultStyleActionBars.GetScrollBtnClass 2978
PlatformDefaultStyleActnCtrls.TPlatformDefaultStyleActionBars.GetStyleName 2978
Play
TMediaPlayer class 2323
pmDouble enumeration member 1264
pmExtended enumeration member 1264
pmReserved enumeration member 1264
pmSingle enumeration member 1264
pngimage 2981
pngimage namespace 2981
Classes 2984
Constants 3029
Functions 3009
Structs, Records, Enums 3020
Types 3023
pngimage.AddtEXt 3009
pngimage.AddtEXt function 3009
pngimage.AddzTXt 3010
pngimage.AddzTXt function 3010
pngimage.Assign 3010
pngimage.Assign function 3010
pngimage.AssignHandle 3010
pngimage.AssignHandle function 3010
pngimage.AssignPNG 3011
pngimage.AssignPNG function 3011
pngimage.COLOR_GRAYSCALE 3029
pngimage.COLOR_GRAYSCALE constant 3029
pngimage.COLOR_GRAYSCALEALPHA 3029
pngimage.COLOR_GRAYSCALEALPHA constant 3029
pngimage.COLOR_PALETTE 3030
pngimage.COLOR_PALETTE constant 3030
pngimage.COLOR_RGB 3030
pngimage.COLOR_RGB constant 3030
pngimage.COLOR_RGBALPHA 3030
pngimage.COLOR_RGBALPHA constant 3030
pngimage.CreateAlpha 3011
pngimage.CreateAlpha function 3011
pngimage.DoSetPalette 3011
pngimage.DoSetPalette function 3011
pngimage.Draw 3012
pngimage.Draw function 3012
pngimage.DrawPartialTrans 3012
pngimage.DrawPartialTrans function 3012
pngimage.DrawUsingPixelInformation 3012
pngimage.DrawUsingPixelInformation function 3012
pngimage.EPNGCannotChangeTransparent 2984
pngimage.EPNGCouldNotLoadResource 2985
pngimage.EPngError 2985
pngimage.EPNGHeaderNotPresent 2985
pngimage.EPNGIHDRNotFirst 2986
pngimage.EPngInvalidCRC 2986
pngimage.EPNGInvalidFileHeader 2987
pngimage.EPngInvalidIHDR 2987
pngimage.EPNGInvalidNewSize 2987
pngimage.EPNGInvalidPalette 2988
pngimage.EPNGInvalidSpec 2988

pngimage.EPNGMissingMultipleIDAT 2988
pngimage.EPNGMissingPalette 2989
pngimage.EPNGNoImageData 2989
pngimage.EPNGNotExists 2990
pngimage.EPNGOutMemory 2990
pngimage.EPNGSizeExceeds 2990
pngimage.EPngUnexpectedEnd 2991
pngimage.EPNGUnknownCompression 2991
pngimage.EPNGUnknownCriticalChunk 2991
pngimage.EPNGUnknownInterlace 2992
pngimage.EPNGZLIBError 2992
pngimage.ExceptClass 3023
pngimage.ExceptClass type 3023
pngimage.Exception 2993
pngimage.Exception.Create 2993
pngimage.FILTER_AVERAGE 3031
pngimage.FILTER_AVERAGE constant 3031
pngimage.FILTER_NONE 3031
pngimage.FILTER_NONE constant 3031
pngimage.FILTER_PAETH 3031
pngimage.FILTER_PAETH constant 3031
pngimage.FILTER_SUB 3032
pngimage.FILTER_SUB constant 3032
pngimage.FILTER_UP 3032
pngimage.FILTER_UP constant 3032
pngimage.GetAlphaScanline 3013
pngimage.GetAlphaScanline function 3013
pngimage.GetEmpty 3013
pngimage.GetEmpty function 3013
pngimage.GetLibraryVersion 3013
pngimage.GetLibraryVersion function 3013
pngimage.GetPalette 3014
pngimage.GetPalette function 3014
pngimage.GetPixelInformation 3014
pngimage.GetPixelInformation function 3014
pngimage.GetPixels 3014
pngimage.GetPixels function 3014
pngimage.GetScanline 3015
pngimage.GetScanline function 3015
pngimage.GetSupportsPartialTransparency 3015
pngimage.GetSupportsPartialTransparency function 3015
pngimage.GetTransparencyMode 3015
pngimage.GetTransparencyMode function 3015
pngimage.GetTransparentColor 3016
pngimage.GetTransparentColor function 3016
pngimage.GetWidth 3016
pngimage.GetWidth function 3016
pngimage.HasPixelInformation 3016
pngimage.HasPixelInformation function 3016
pngimage.Index 3032
pngimage.Index constant 3032
pngimage.LibraryVersion 3033
pngimage.LibraryVersion constant 3033
pngimage.pByteArray 3023
pngimage.pByteArray type 3023
pngimage.pCardinal 3024
pngimage.pCardinal type 3024
pngimage.pPointerArray 3024
pngimage.pPointerArray type 3024
pngimage.pRGBLine 3024
pngimage.pRGBLine type 3024
pngimage.pRGBPixel 3025
pngimage.pRGBPixel type 3025
pngimage.RaiseError 3017
pngimage.RaiseError function 3017
pngimage.RemoveTransparency 3017
pngimage.RemoveTransparency function 3017
pngimage.Resize 3017
pngimage.Resize function 3017
pngimage.SetMaxIdatSize 3018
pngimage.SetMaxIdatSize function 3018
pngimage.SetPixels 3018
pngimage.SetPixels function 3018
pngimage.SetTransparentColor 3018
pngimage.SetTransparentColor function 3018
pngimage.SetWidth 3019
pngimage.SetWidth function 3019
pngimage.soFromBeginning 3033

pngimage.soFromBeginning constant 3033
pngimage.soFromCurrent 3033
pngimage.soFromCurrent constant 3033
pngimage.soFromEnd 3034
pngimage.soFromEnd constant 3034
pngimage.TBitmap 3025
pngimage.TBitmap type 3025
pngimage.TByteArray 3025
pngimage.TByteArray type 3025
pngimage.TCanvas 3026
pngimage.TCanvas type 3026
pngimage.TChunkClass 3026
pngimage.TChunkClass type 3026
pngimage.TColor 3026
pngimage.TColor type 3026
pngimage.TCompressionLevel 3027
pngimage.TCompressionLevel type 3027
pngimage.TFileStream 2993
pngimage.TFileStream.Create 2994
pngimage.TFileStream.Destroy 2994
pngimage.TFileStream.Read 2995
pngimage.TFileStream.Seek 2995
pngimage.TFileStream.SetSize 2995
pngimage.TFileStream.Write 2996
pngimage.TFileStreamMode 3020
pngimage.TFileStreamMode enumeration 3020
pngimage.TFileStreamModeSet 3027
pngimage.TFileStreamModeSet type 3027
pngimage.TFilter 3020
pngimage.TFilter enumeration 3020
pngimage.TFilters 3027
pngimage.TFilters type 3027
pngimage.TInterlaceMethod 3021
pngimage.TInterlaceMethod enumeration 3021
pngimage.TMAXBITMAPINFO 3021
pngimage.TMAXBITMAPINFO record 3021
pngimage.TPersistent 3028
pngimage.TPersistent type 3028
pngimage.TPngImage 2996
pngimage.TPNGList 2997
pngimage.TPNGList.Add 2998
pngimage.TPNGList.FindChunk 2998
pngimage.TPNGList.Item 2999
pngimage.TPNGList.ItemFromClass 2998
pngimage.TPNGList.RemoveChunk 2999
pngimage.TPNGPointerList 2999
pngimage.TPNGPointerList.Add 3001
pngimage.TPNGPointerList.Count 3002
pngimage.TPNGPointerList.Create 3000
pngimage.TPNGPointerList.Destroy 3000
pngimage.TPNGPointerList.Insert 3001
pngimage.TPNGPointerList.Item 3002
pngimage.TPNGPointerList.Owner 3002
pngimage.TPNGPointerList.Remove 3001
pngimage.TPNGPointerList.SetSize 3001
pngimage.TPNGTransparencyMode 3022
pngimage.TPNGTransparencyMode enumeration 3022
pngimage.TPointerArray 3028
pngimage.TPointerArray type 3028
pngimage.TResourceStream 3003
pngimage.TResourceStream.Create 3003
pngimage.TResourceStream.Read 3004
pngimage.TResourceStream.Seek 3004
pngimage.TResourceStream.SetSize 3004
pngimage.TResourceStream.Write 3005
pngimage.TRGBLine 3028
pngimage.TRGBLine type 3028
pngimage.TRGBPixel 3022
pngimage.TRGBPixel record 3022
pngimage.TStream 3005
pngimage.TStream.CopyFrom 3006
pngimage.TStream.GetPosition 3006
pngimage.TStream.GetSize 3006
pngimage.TStream.Position 3008
pngimage.TStream.Read 3007
pngimage.TStream.Seek 3007
pngimage.TStream.SetPosition 3007
pngimage.TStream.SetSize 3007

pngimage.TStream.Size 3008
pngimage.TStream.Write 3007
pngimage.Z_FINISH 3034
pngimage.Z_FINISH constant 3034
pngimage.Z_NO_FLUSH 3034
pngimage.Z_NO_FLUSH constant 3034
pngimage.Z_STREAM_END 3035
pngimage.Z_STREAM_END constant 3035
pngimage.ZLIBErrors 3035
pngimage.ZLIBErrors constant 3035
poAllowCommandText enumeration member 3218
poAllowMultiRecordUpdates enumeration member 3218
poAutoRefresh enumeration member 3218
poCascadeDeletes enumeration member 3218
poCascadeUpdates enumeration member 3218
poDisableDeletes enumeration member 3218
poDisableEdits enumeration member 3218
poDisableInserts enumeration member 3218
poFetchBlobsOnDemand enumeration member 3218
poFetchDetailsOnDemand enumeration member 3218
poHorizontal enumeration member 707
poIncFieldProps enumeration member 3218
poLandscape enumeration member 3070
poNoReset enumeration member 3218
Pooled
TMtsAutoObject class 2394
TMtsDataModule class 2412
poPortrait enumeration member 3070
poPropogateChanges enumeration member 3218
PopulateOle2Menu
TMainMenu class 1326
Popup
TPopupMenu class 1434
PopupComponent
TPopupMenu class 1440
PopupMenu
TMaskEdit class 1128
TMediaPlayer class 2352
TOleContainer class 2571
TOutline class 2896
TRibbonComboBox class 3637
TRibbonSpinButton class 3810
TRibbonSpinEdit class 3856
PopupPoint
TPopupMenu class 1443
poReadOnly enumeration member 3218
poRetainServerOrder enumeration member 3218
Port
TServerItem class 2474
Position
TMediaPlayer class 2357
TQuickAccessToolbar class 3474
TStream class 3008
PositionPopup
TRibbonApplicationPopupMenu class 3967
TRibbonQATCustomizeMenu class 3970
TRibbonStyleMainPopupMenu class 3976
Post
TIBCustomDataSet class 86
PostClick
TMediaPlayer class 2326
PostMessageFilter
TOleControl class 2674
poUseQuoteChar enumeration member 3218
poVertical enumeration member 707
PreMessageFilter
TOleControl class 2674
Prepare
TIBDataSet class 120
TIBQuery class 385
TIBSQL class 528
TIBStoredProc class 590
Prepared
TIBDataSet class 131
TIBQuery class 391
TIBSQL class 533
TIBStoredProc class 594
PreprocessSQL

TIBSQL class 528
Previous
TMediaPlayer class 2327
PrimaryVerb
TOleContainer class 2576
PrinterIndex
TPrinter class 3062
Printers 3037
TPrinter class 3062
Printers namespace 3037
Classes 3038
Functions 3066
Structs, Records, Enums 3070
Types 3072
Printers.AssignPrn 3066
Printers.AssignPrn function 3066
Printers.EPrinter 3038
Printers.Printer 3066
Printers.Printer function 3066
Printers.SetPrinter 3068
Printers.SetPrinter function 3068
Printers.TPrinter 3038
Printers.TPrinter.Abort 3041
Printers.TPrinter.Aborted 3051
Printers.TPrinter.BeginDoc 3043
Printers.TPrinter.Canvas 3053
Printers.TPrinter.Capabilities 3053
Printers.TPrinter.Copies 3054
Printers.TPrinter.Create 3040
Printers.TPrinter.Destroy 3040
Printers.TPrinter.EndDoc 3045
Printers.TPrinter.Fonts 3054
Printers.TPrinter.GetPrinter 3047
Printers.TPrinter.Handle 3054
Printers.TPrinter.NewPage 3047
Printers.TPrinter.Orientation 3056
Printers.TPrinter.PageHeight 3057
Printers.TPrinter.PageNumber 3059
Printers.TPrinter.PageWidth 3060
Printers.TPrinter.PrinterIndex 3062
Printers.TPrinter.Printers 3062
Printers.TPrinter.Printing 3062
Printers.TPrinter.Refresh 3049
Printers.TPrinter.SetPrinter 3050
Printers.TPrinter.Title 3065
Printers.TPrinterCapabilities 3072
Printers.TPrinterCapabilities type 3072
Printers.TPrinterCapability 3070
Printers.TPrinterCapability enumeration 3070
Printers.TPrinterOrientation 3070
Printers.TPrinterOrientation enumeration 3070
Printers.TPrinterState 3071
Printers.TPrinterState enumeration 3071
Printing
TPrinter class 3062
PrintSet
TIBExtract class 280
ProblemCount
TIBBatchMove class 35
ProblemTableName
TIBBatchMove class 35
ProcessMenuChar
TMenu class 1342
ProcessResponse
IOPConvert interface 2736
TOPToSoapDomConvert class 2755
Progress
TIBSetup class 320
ProgressiveDisplay
TJPEGImage class 1009
ProgressiveEncoding
TJPEGImage class 1010
PropertyByName
ISharedPropertyGroup interface 2431
PropertyByPosition
ISharedPropertyGroup interface 2431
Protocol
TIBCustomService class 431

Provider 3073
TCustomResolver class 3135
TDataSetResolver class 3182
TSQLResolver class 3204
Provider namespace 3073
Classes 3075
Constants 3232
Functions 3215
Interfaces 3213
Structs, Records, Enums 3217
Types 3222
Variables 3231
Provider.EDSWriter 3075
Provider.EDSWriter.Create 3076
Provider.EDSWriter.ErrorCode 3076
Provider.GetObjectProperty 3215
Provider.GetObjectProperty function 3215
Provider.GetStringProperty 3215
Provider.GetStringProperty function 3215
Provider.InformixLob 3231
Provider.InformixLob variable 3231
Provider.IProviderContainer 3213
Provider.IProviderContainer.RegisterProvider 3214
Provider.IProviderContainer.UnRegisterProvider 3214
Provider.MetaDataOption 3232
Provider.MetaDataOption constant 3232
Provider.PPutFieldInfo 3222
Provider.PPutFieldInfo type 3222
Provider.ResetOption 3233
Provider.ResetOption constant 3233
Provider.TAfterUpdateRecordEvent 3223
Provider.TAfterUpdateRecordEvent type 3223
Provider.TBaseProvider 3077
Provider.TBaseProvider.AfterUpdateRecord 3090
Provider.TBaseProvider.BeforeUpdateRecord 3090
Provider.TBaseProvider.CheckResolver 3081
Provider.TBaseProvider.Create 3079
Provider.TBaseProvider.CreateDataPacket 3081
Provider.TBaseProvider.CreateResolver 3082
Provider.TBaseProvider.Destroy 3080
Provider.TBaseProvider.DoAfterUpdateRecord 3082
Provider.TBaseProvider.DoBeforeCommit 3082
Provider.TBaseProvider.DoBeforeUpdateRecord 3083
Provider.TBaseProvider.DoOnGetData 3083
Provider.TBaseProvider.DoOnUpdateData 3083
Provider.TBaseProvider.FetchDetails 3083
Provider.TBaseProvider.FreeResolver 3084
Provider.TBaseProvider.InternalApplyUpdates 3084
Provider.TBaseProvider.InternalGetRecords 3085
Provider.TBaseProvider.InternalRowRequest 3086
Provider.TBaseProvider.LocateRecord 3086
Provider.TBaseProvider.OnDataRequest 3091
Provider.TBaseProvider.OnGetData 3091
Provider.TBaseProvider.OnUpdateData 3092
Provider.TBaseProvider.OnUpdateError 3092
Provider.TBaseProvider.Options 3087
Provider.TBaseProvider.PacketDataSet 3088
Provider.TBaseProvider.Resolver 3088
Provider.TBaseProvider.UpdateMode 3089
Provider.TBaseProvider.UpdateRecord 3087
Provider.TBeforeCommitEvent 3223
Provider.TBeforeCommitEvent type 3223
Provider.TBeforeUpdateRecordEvent 3224
Provider.TBeforeUpdateRecordEvent type 3224
Provider.TCustomPacketWriter 3093
Provider.TCustomPacketWriter.AddAttribute 3095
Provider.TCustomPacketWriter.Check 3095
Provider.TCustomPacketWriter.Create 3094
Provider.TCustomPacketWriter.Destroy 3094
Provider.TCustomPacketWriter.DSWriter 3096
Provider.TCustomPacketWriter.FBuffer 3095
Provider.TCustomPacketWriter.SetDSWriter 3096
Provider.TCustomProvider 3096
Provider.TCustomProvider.ActiveUpdateException 3114
Provider.TCustomProvider.AfterApplyUpdates 3116
Provider.TCustomProvider.AfterExecute 3117
Provider.TCustomProvider.AfterGetParams 3118
Provider.TCustomProvider.AfterGetRecords 3118

Provider.TCustomProvider.AfterRowRequest 3119
Provider.TCustomProvider.ApplyUpdates 3100
Provider.TCustomProvider.BeforeApplyUpdates 3120
Provider.TCustomProvider.BeforeExecute 3121
Provider.TCustomProvider.BeforeGetParams 3121
Provider.TCustomProvider.BeforeGetRecords 3122
Provider.TCustomProvider.BeforeRowRequest 3123
Provider.TCustomProvider.Create 3099
Provider.TCustomProvider.Data 3115
Provider.TCustomProvider.DataRequest 3101
Provider.TCustomProvider.Destroy 3099
Provider.TCustomProvider.DoAfterApplyUpdates 3102
Provider.TCustomProvider.DoAfterExecute 3102
Provider.TCustomProvider.DoAfterGetParams 3102
Provider.TCustomProvider.DoAfterGetRecords 3103
Provider.TCustomProvider.DoAfterRowRequest 3103
Provider.TCustomProvider.DoApplyUpdates 3104
Provider.TCustomProvider.DoBeforeApplyUpdates 3104
Provider.TCustomProvider.DoBeforeExecute 3105
Provider.TCustomProvider.DoBeforeGetParams 3105
Provider.TCustomProvider.DoBeforeGetRecords 3106
Provider.TCustomProvider.DoBeforeRowRequest 3106
Provider.TCustomProvider.DoGetRecords 3107
Provider.TCustomProvider.Execute 3107
Provider.TCustomProvider.Exported 3115
Provider.TCustomProvider.GetParams 3108
Provider.TCustomProvider.GetRecords 3109
Provider.TCustomProvider.InternalApplyUpdates 3110
Provider.TCustomProvider.InternalExecute 3111
Provider.TCustomProvider.InternalGetParams 3111
Provider.TCustomProvider.InternalGetRecords 3112
Provider.TCustomProvider.InternalRowRequest 3113
Provider.TCustomProvider.OnDataRequest 3124
Provider.TCustomProvider.OnValidate 3125
Provider.TCustomProvider.OwnerData 3115
Provider.TCustomProvider.RowRequest 3113
Provider.TCustomProvider.SetActiveUpdateException 3114
Provider.TCustomResolver 3125
Provider.TCustomResolver.ApplyUpdates 3128
Provider.TCustomResolver.BeginUpdate 3128
Provider.TCustomResolver.Create 3126
Provider.TCustomResolver.Destroy 3127
Provider.TCustomResolver.DoDelete 3129
Provider.TCustomResolver.DoInsert 3129
Provider.TCustomResolver.DoUpdate 3129
Provider.TCustomResolver.EndUpdate 3130
Provider.TCustomResolver.FreeTreeData 3130
Provider.TCustomResolver.HandleUpdateError 3131
Provider.TCustomResolver.InitializeConflictBuffer 3131
Provider.TCustomResolver.InitKeyFields 3132
Provider.TCustomResolver.InitTreeData 3132
Provider.TCustomResolver.InternalBeforeResolve 3133
Provider.TCustomResolver.InternalUpdateRecord 3133
Provider.TCustomResolver.LogUpdateError 3133
Provider.TCustomResolver.LogUpdateRecord 3134
Provider.TCustomResolver.Provider 3135
Provider.TCustomResolver.RowRequest 3134
Provider.TCustomResolver.UpdateTree 3135
Provider.TDataPacketWriter 3136
Provider.TDataPacketWriter.AddColumn 3138
Provider.TDataPacketWriter.AddConstraints 3138
Provider.TDataPacketWriter.AddDataSetAttributes 3139
Provider.TDataPacketWriter.AddExtraFieldProps 3139
Provider.TDataPacketWriter.AddFieldLinks 3139
Provider.TDataPacketWriter.AddIndexDefs 3139
Provider.TDataPacketWriter.Constraints 3144
Provider.TDataPacketWriter.Destroy 3137
Provider.TDataPacketWriter.FreeInfoRecords 3140
Provider.TDataPacketWriter.GetDataPacket 3140
Provider.TDataPacketWriter.GetFieldIdx 3140
Provider.TDataPacketWriter.InitPutProcs 3140
Provider.TDataPacketWriter.OnGetParams 3144
Provider.TDataPacketWriter.Options 3144
Provider.TDataPacketWriter.PacketOptions 3144
Provider.TDataPacketWriter.PutADTField 3140
Provider.TDataPacketWriter.PutArrayField 3141
Provider.TDataPacketWriter.PutBlobField 3141
Provider.TDataPacketWriter.PutCalcField 3141

Provider.TDataPacketWriter.PutDataSetField 3141
Provider.TDataPacketWriter.PutField 3142
Provider.TDataPacketWriter.PutStringField 3142
Provider.TDataPacketWriter.PutVarBytesField 3142
Provider.TDataPacketWriter.PutWideStringField 3142
Provider.TDataPacketWriter.RefreshPutProcs 3142
Provider.TDataPacketWriter.Reset 3143
Provider.TDataPacketWriter.WriteDataSet 3143
Provider.TDataPacketWriter.WriteMetaData 3143
Provider.TDataRequestEvent 3224
Provider.TDataRequestEvent type 3224
Provider.TDataSetProvider 3145
Provider.TDataSetProvider.ActiveUpdateException 3159
Provider.TDataSetProvider.AfterApplyUpdates 3164
Provider.TDataSetProvider.AfterExecute 3165
Provider.TDataSetProvider.AfterGetParams 3166
Provider.TDataSetProvider.AfterGetRecords 3166
Provider.TDataSetProvider.AfterRowRequest 3167
Provider.TDataSetProvider.AfterUpdateRecord 3168
Provider.TDataSetProvider.BeforeApplyUpdates 3168
Provider.TDataSetProvider.BeforeExecute 3169
Provider.TDataSetProvider.BeforeGetParams 3170
Provider.TDataSetProvider.BeforeGetRecords 3170
Provider.TDataSetProvider.BeforeRowRequest 3172
Provider.TDataSetProvider.BeforeUpdateRecord 3172
Provider.TDataSetProvider.CheckDataSet 3149
Provider.TDataSetProvider.Constraints 3159
Provider.TDataSetProvider.Create 3148
Provider.TDataSetProvider.CreateDataPacket 3149
Provider.TDataSetProvider.CreateResolver 3150
Provider.TDataSetProvider.DataSet 3160
Provider.TDataSetProvider.Destroy 3148
Provider.TDataSetProvider.DoBeforeExecute 3150
Provider.TDataSetProvider.DoBeforeGetRecords 3151
Provider.TDataSetProvider.DoGetProviderAttributes 3152
Provider.TDataSetProvider.DoGetTableName 3152
Provider.TDataSetProvider.Exported 3160
Provider.TDataSetProvider.FetchDetails 3153
Provider.TDataSetProvider.FindRecord 3153
Provider.TDataSetProvider.GetDataSetFromDelta 3153
Provider.TDataSetProvider.InternalApplyUpdates 3154
Provider.TDataSetProvider.InternalExecute 3154
Provider.TDataSetProvider.InternalGetParams 3155
Provider.TDataSetProvider.InternalGetRecords 3155
Provider.TDataSetProvider.InternalRowRequest 3156
Provider.TDataSetProvider.LocateRecord 3157
Provider.TDataSetProvider.Notification 3157
Provider.TDataSetProvider.OnDataRequest 3173
Provider.TDataSetProvider.OnGetData 3174
Provider.TDataSetProvider.OnGetDataSetProperties 3174
Provider.TDataSetProvider.OnGetTableName 3175
Provider.TDataSetProvider.OnUpdateData 3175
Provider.TDataSetProvider.OnUpdateError 3176
Provider.TDataSetProvider.OnValidate 3176
Provider.TDataSetProvider.Options 3161
Provider.TDataSetProvider.OwnerData 3162
Provider.TDataSetProvider.Params 3162
Provider.TDataSetProvider.Reset 3157
Provider.TDataSetProvider.ResolveToDataSet 3162
Provider.TDataSetProvider.SetCommandText 3158
Provider.TDataSetProvider.SetParams 3158
Provider.TDataSetProvider.TransactionStarted 3163
Provider.TDataSetProvider.UpdateMode 3163
Provider.TDataSetProvider.UpdateRecord 3158
Provider.TDataSetResolver 3176
Provider.TDataSetResolver.BeginUpdate 3178
Provider.TDataSetResolver.Create 3178
Provider.TDataSetResolver.DoDelete 3179
Provider.TDataSetResolver.DoInsert 3179
Provider.TDataSetResolver.DoUpdate 3180
Provider.TDataSetResolver.EndUpdate 3180
Provider.TDataSetResolver.InitializeConflictBuffer 3181
Provider.TDataSetResolver.InternalBeforeResolve 3181
Provider.TDataSetResolver.Provider 3182
Provider.TGetDSProps 3225
Provider.TGetDSProps type 3225
Provider.TGetParamsEvent 3225
Provider.TGetParamsEvent type 3225

Provider.TGetRecordOption 3217
Provider.TGetRecordOption enumeration 3217
Provider.TGetRecordOptions 3226
Provider.TGetRecordOptions type 3226
Provider.TGetTableNameEvent 3226
Provider.TGetTableNameEvent type 3226
Provider.TInfoArray 3227
Provider.TInfoArray type 3227
Provider.TLocalAppServer 3182
Provider.TLocalAppServer.AS_ApplyUpdates 3185
Provider.TLocalAppServer.AS_DataRequest 3185
Provider.TLocalAppServer.AS_Execute 3185
Provider.TLocalAppServer.AS_GetParams 3185
Provider.TLocalAppServer.AS_GetProviderNames 3186
Provider.TLocalAppServer.AS_GetRecords 3186
Provider.TLocalAppServer.AS_RowRequest 3186
Provider.TLocalAppServer.Create 3183
Provider.TLocalAppServer.Destroy 3184
Provider.TLocalAppServer.GetIDsOfNames 3186
Provider.TLocalAppServer.GetTypeInfo 3187
Provider.TLocalAppServer.GetTypeInfoCount 3187
Provider.TLocalAppServer.InterfaceSupportsErrorInfo 3187
Provider.TLocalAppServer.Invoke 3187
Provider.TLocalAppServer.SafeCallException 3188
Provider.TOnValidate 3227
Provider.TOnValidate type 3227
Provider.TPacketDataSet 3188
Provider.TPacketDataSet.AssignCurValues 3190
Provider.TPacketDataSet.Create 3189
Provider.TPacketDataSet.CreateFromDelta 3191
Provider.TPacketDataSet.DataEvent 3191
Provider.TPacketDataSet.GetStateFieldValue 3191
Provider.TPacketDataSet.HasCurValues 3192
Provider.TPacketDataSet.HasMergeConflicts 3192
Provider.TPacketDataSet.InitAltRecBuffers 3192
Provider.TPacketDataSet.InternalClose 3193
Provider.TPacketDataSet.InternalInitRecord 3193
Provider.TPacketDataSet.InternalOpen 3193
Provider.TPacketDataSet.NewValuesModified 3195
Provider.TPacketDataSet.SetFieldData 3194
Provider.TPacketDataSet.SetStateFieldValue 3194
Provider.TPacketDataSet.StreamMetaData 3195
Provider.TPacketDataSet.UpdateKind 3194
Provider.TPacketDataSet.UseCurValues 3195
Provider.TPacketDataSet.WritingCurValues 3196
Provider.TProvider 3196
Provider.TProviderDataEvent 3228
Provider.TProviderDataEvent type 3228
Provider.TProviderOption 3218
Provider.TProviderOption enumeration 3218
Provider.TProviderOptions 3228
Provider.TProviderOptions type 3228
Provider.TPutFieldInfo 3220
Provider.TPutFieldInfo record 3220
Provider.TPutFieldProc 3229
Provider.TPutFieldProc type 3229
Provider.TResolverErrorEvent 3230
Provider.TResolverErrorEvent type 3230
Provider.TResolverResponse 3221
Provider.TResolverResponse enumeration 3221
Provider.TSQLResolver 3196
Provider.TSQLResolver.Create 3198
Provider.TSQLResolver.Destroy 3198
Provider.TSQLResolver.DoDelete 3199
Provider.TSQLResolver.DoExecSQL 3200
Provider.TSQLResolver.DoGetValues 3200
Provider.TSQLResolver.DoInsert 3200
Provider.TSQLResolver.DoUpdate 3201
Provider.TSQLResolver.FreeTreeData 3201
Provider.TSQLResolver.GenDeleteSQL 3201
Provider.TSQLResolver.GenInsertSQL 3202
Provider.TSQLResolver.GenSelectSQL 3202
Provider.TSQLResolver.GenUpdateSQL 3202
Provider.TSQLResolver.GenWhereSQL 3202
Provider.TSQLResolver.GetProvider 3203
Provider.TSQLResolver.InitializeConflictBuffer 3203
Provider.TSQLResolver.InitTreeData 3203
Provider.TSQLResolver.InternalDoUpdate 3204

Provider.TSQLResolver.Provider 3204
Provider.TSQLResolver.UseFieldInUpdate 3204
Provider.TSQLResolver.UseFieldInWhere 3204
Provider.TUpdateTree 3205
Provider.TUpdateTree.Clear 3207
Provider.TUpdateTree.Create 3206
Provider.TUpdateTree.Data 3209
Provider.TUpdateTree.Delta 3210
Provider.TUpdateTree.Destroy 3207
Provider.TUpdateTree.DetailCount 3210
Provider.TUpdateTree.DetailList 3210
Provider.TUpdateTree.Details 3210
Provider.TUpdateTree.DoUpdates 3207
Provider.TUpdateTree.ErrorDS 3211
Provider.TUpdateTree.HasErrors 3211
Provider.TUpdateTree.InitData 3208
Provider.TUpdateTree.InitDelta 3208
Provider.TUpdateTree.InitErrorPacket 3209
Provider.TUpdateTree.IsNested 3211
Provider.TUpdateTree.Name 3211
Provider.TUpdateTree.Opened 3211
Provider.TUpdateTree.Parent 3212
Provider.TUpdateTree.RefreshData 3209
Provider.TUpdateTree.Resolver 3212
Provider.TUpdateTree.Source 3212
Provider.VarArrayFromStrings 3215
Provider.VarArrayFromStrings function 3215
Provider.VarArrayFromWideStrings 3216
Provider.VarArrayFromWideStrings function 3216
Provider.XMLOption 3233
Provider.XMLOption constant 3233
Provider.XMLUTF8Option 3234
Provider.XMLUTF8Option constant 3234
PSEndTransaction
TIBCustomDataSet class 86
PSExecute
TIBQuery class 385
TIBStoredProc class 590
PSExecuteStatement
TIBCustomDataSet class 86
PSGetDefaultOrder
TIBTable class 612
PSGetIndexDefs
TIBTable class 612
PSGetKeyFields
TIBTable class 612
PSGetParams
TIBQuery class 385
TIBStoredProc class 590
PSGetQuoteChar
TIBCustomDataSet class 87
PSGetTableName
TIBCustomDataSet class 87
TIBQuery class 385
TIBStoredProc class 590
TIBTable class 612
PSGetUpdateException
TIBCustomDataSet class 87
psHandleDC enumeration member 3071
psHandleIC enumeration member 3071
PSInTransaction
TIBCustomDataSet class 87
PSIsSQLBased
TIBCustomDataSet class 88
PSIsSQLSupported
TIBCustomDataSet class 88
psNoHandle enumeration member 3071
PSReset
TIBCustomDataSet class 88
PSSetCommandText
TIBDataSet class 120
TIBQuery class 386
TIBStoredProc class 591
TIBTable class 613
PSSetParams
TIBQuery class 386
TIBStoredProc class 591
TIBTable class 613

PSStartTransaction
TIBCustomDataSet class 88
PSUpdateRecord
TIBCustomDataSet class 88
ptEndOfPeriod enumeration member 1266
ptmBit enumeration member 3022
ptmNone enumeration member 3022
ptmPartial enumeration member 3022
ptStartOfPeriod enumeration member 1266
PurgeCount
TIBDatabaseInfo class 253
PutADTField
TDataPacketWriter class 3140
PutAny
IMarshalOutBuffer interface 2802
PutArrayField
TDataPacketWriter class 3141
PutBlobField
TDataPacketWriter class 3141
PutCalcField
TDataPacketWriter class 3141
PutChar
IMarshalOutBuffer interface 2802
PutData
TRegistry class 3273
PutDataSetField
TDataPacketWriter class 3141
PutDouble
IMarshalOutBuffer interface 2802
PutField
TDataPacketWriter class 3142
PutFloat
IMarshalOutBuffer interface 2802
PutLong
IMarshalOutBuffer interface 2803
PutObject
IMarshalOutBuffer interface 2803
PutShort
IMarshalOutBuffer interface 2803
PutStringField
TDataPacketWriter class 3142
PutText
IMarshalOutBuffer interface 2803
PutUnsignedChar
IMarshalOutBuffer interface 2804
PutUnsignedLong
IMarshalOutBuffer interface 2804
PutUnsignedShort
IMarshalOutBuffer interface 2804
PutVarBytesField
TDataPacketWriter class 3142
PutWidechar
IMarshalOutBuffer interface 2804
PutWideStringField
TDataPacketWriter class 3142
PutWideText
IMarshalOutBuffer interface 2804

Q

QDelete
TIBCustomDataSet class 103
QInsert
TIBCustomDataSet class 104
QModify
TIBCustomDataSet class 104
qpBottom enumeration member 3471
qpTop enumeration member 3471
QRefresh
TIBCustomDataSet class 104
qsCustomize enumeration member 3929
QSelect
TIBCustomDataSet class 104
qsMoreControls enumeration member 3929
Query
TIBUpdateSQL class 636
QueryInterface
TAutoDispatch class 2503
TEventDispatch class 2638

TInvokableClass class 839
TOleContainer class 2547
TOleControl class 2675
TOleServer class 2716
TRIO class 4063
TServerEventDispatch class 2723
TXMLDataSet class 2169
QueryService
TOleControl class 2675
QuickAccessToolbar
TCustomRibbon class 3397
TRibbon class 3496
QuickAccessToolbarMoreBorderColor
TCustomRibbonColorMap class 3405
TRibbonLunaColorMap class 3530
TRibbonObsidianColorMap class 3536
TRibbonSilverColorMap class 3554
QuickAccessToolbarMoreColor
TCustomRibbonColorMap class 3405
TRibbonLunaColorMap class 3530
TRibbonObsidianColorMap class 3536
TRibbonSilverColorMap class 3554

R

RadioItem
TMenuItem class 1414
rdBinary enumeration member 3327
rdExpandString enumeration member 3327
rdInteger enumeration member 3327
rdString enumeration member 3327
rdUnknown enumeration member 3327
Read
TFileStream class 2995
TIBBlobStream class 47
TIBDSBlobStream class 140
TResourceStream class 3004
TStream class 3007
ReadBinaryData
TRegistry class 3273
ReadBinaryStream
TCustomIniFile class 722
TRegistryIniFile class 3306
ReadBlanksAsNull
TIBInputDelimitedFile class 504
ReadBool
TCustomIniFile class 723
TRegIniFile class 3241
TRegistry class 3274
ReadClient
TISAPIRequest class 974
ReadCurrency
TRegistry class 3274
ReadData
TCustomImageList class 672
TJPEGImage class 1004
ReadDate
TCustomIniFile class 726
TRegistry class 3274
TRegistryIniFile class 3307
ReadDateTime
TCustomIniFile class 726
TRegistry class 3275
TRegistryIniFile class 3307
ReadFloat
TCustomIniFile class 727
TRegistry class 3275
TRegistryIniFile class 3308
ReadHeader
TSOAPDomConv class 2775
ReadIdxCount
TIBDatabaseInfo class 253
ReadInteger
TCustomIniFile class 727
TRegIniFile class 3242
TRegistry class 3276
TRegistryIniFile class 3308
ReadOnly
TFieldRadioGroup class 1882

TFieldText class 1904
TFieldTextArea class 1911
TIBDatabaseInfo class 253
TIBTable class 621
TMaskEdit class 1133
TQueryRadioGroup class 1986
TQueryText class 2008
TQueryTextArea class 2020
TRibbonComboBox class 3642
TRibbonSpinEdit class 3861
TTextAreaColumn class 2046
TTextColumn class 2054
TWebRadioGroupInput class 2128
TWebTextAreaInput class 2144
TWebTextInput class 2150
ReadParameters
TIBBatchInput class 498
TIBInputDelimitedFile class 503
TIBInputRawFile class 507
ReadRectDynArray
TSOAPDomConv class 2776
ReadRectDynArrayElem
TSOAPDomConv class 2776
ReadRow
TSOAPDomConv class 2776
Reads
TIBDatabaseInfo class 253
ReadSection
TCustomIniFile class 730
TIniFile class 758
TMemIniFile class 773
TRegIniFile class 3242
TRegistryIniFile class 3311
ReadSections
TCustomIniFile class 731
TIniFile class 758
TMemIniFile class 773
TRegIniFile class 3243
TRegistryIniFile class 3312
ReadSectionValues
TCustomIniFile class 732
TIniFile class 759
TMemIniFile class 774
TRegIniFile class 3243
TRegistryIniFile class 3313
ReadSeqCount
TIBDatabaseInfo class 254
ReadStream
TJPEGImage class 1005
ReadString
TCustomIniFile class 732
TIniFile class 759
TISAPIRequest class 974
TMemIniFile class 774
TRegIniFile class 3244
TRegistry class 3276
TRegistryIniFile class 3313
ReadTime
TCustomIniFile class 735
TRegistry class 3277
TRegistryIniFile class 3316
ReadVarArrayDim
TSOAPDomConv class 2776
ReadVariant
TSOAPDomConv class 2777
ReadyFile
TIBBatch class 496
TIBInputDelimitedFile class 503
TIBInputRawFile class 507
TIBOutputDelimitedFile class 510
TIBOutputRawFile class 513
reApplicationMenu enumeration member 4029
RebootToComplete
TIBSetup class 320
RebuildRibbonMetrics
TCustomRibbon class 3387
reButton enumeration member 4029
RecentItems

TRibbonApplicationMenuBar class 3956
reCheckBox enumeration member 4029
reCombo enumeration member 4029
RecordChanged
TIBDataLink class 114
RecordCount
TIBBatchMove class 36
TIBSQL class 533
RecordModified
TIBCustomDataSet class 89
RecordSize
TIBXSQLDA class 543
RecoverTwoPhaseGlobal enumeration member 483
RefCount
TAutoObject class 2511
TInvokableClass class 839
TRIO class 4064
reForm enumeration member 4029
ReformatText
TCustomMaskEdit class 1089
Refresh
TPrinter class 3049
RefreshControl
TListActionLink class 1040
RefreshData
TUpdateTree class 3209
RefreshPutProcs
TDataPacketWriter class 3142
RefreshSQL
TIBCustomDataSet class 105
TIBDataSet class 131
TIBDataSetUpdateObject class 137
reGallery enumeration member 4029
RegIniFile
TRegistryIniFile class 3326
RegisterAllSOAPActions
TInvokableClassRegistry class 859
RegisterChanges
TCustomImageList class 672
RegisterClass
TAutomation class 2505
RegisterDefaultSOAPAction
TInvokableClassRegistry class 859
Registered
TIBEvents class 264
RegisterEvents
IIBEventNotifier interface 222
TIBEvents class 261
RegisterException
TInvokableClassRegistry class 860
RegisterExceptionMethod
TInvokableClassRegistry class 860
RegisterExternalMethName
TInvokableClassRegistry class 861
RegisterExternalParamName
TInvokableClassRegistry class 861
RegisterExternalPropName
TRemotableTypeRegistry class 883
RegisterHeaderClass
TInvokableClassRegistry class 862
RegisterHeaderMethod
TInvokableClassRegistry class 863
RegisterHolderClsMember
TRemotableTypeRegistry class 883
RegisterInterface
TInvokableClassRegistry class 864
RegisterInvokableClass
TInvokableClassRegistry class 865
RegisterInvokeOptions
TInvokableClassRegistry class 866
RegisterMonitor
IIBSQLMonitorHook interface 570
RegisterProvider
IProviderContainer interface 3214
RegisterReturnParamNames
TInvokableClassRegistry class 867
RegisterSerializeOptions
TRemotableTypeRegistry class 884

RegisterUDDIInfo
TInvokableClassRegistry class 867
RegisterXSClass
TRemotableTypeRegistry class 885
RegisterXSInfo
TRemotableTypeRegistry class 886
Registry 3235
Registry namespace 3235
Classes 3236
Structs, Records, Enums 3327
Registry.ERegistryException 3236
Registry.TRegDataInfo 3327
Registry.TRegDataInfo record 3327
Registry.TRegDataType 3327
Registry.TRegDataType enumeration 3327
Registry.TRegIniFile 3237
Registry.TRegIniFile.Create 3239
Registry.TRegIniFile.DeleteKey 3240
Registry.TRegIniFile.EraseSection 3241
Registry.TRegIniFile.FileName 3246
Registry.TRegIniFile.ReadBool 3241
Registry.TRegIniFile.ReadInteger 3242
Registry.TRegIniFile.ReadSection 3242
Registry.TRegIniFile.ReadSections 3243
Registry.TRegIniFile.ReadSectionValues 3243
Registry.TRegIniFile.ReadString 3244
Registry.TRegIniFile.WriteBool 3244
Registry.TRegIniFile.WriteInteger 3245
Registry.TRegIniFile.WriteString 3245
Registry.TRegistry 3246
Registry.TRegistry.Access 3293
Registry.TRegistry.ChangeKey 3255
Registry.TRegistry.CloseKey 3255
Registry.TRegistry.Create 3251
Registry.TRegistry.CreateKey 3258
Registry.TRegistry.CurrentKey 3296
Registry.TRegistry.CurrentPath 3296
Registry.TRegistry.DeleteKey 3259
Registry.TRegistry.DeleteValue 3261
Registry.TRegistry.Destroy 3253
Registry.TRegistry.GetBaseKey 3262
Registry.TRegistry.GetData 3262
Registry.TRegistry.GetDataAsString 3262
Registry.TRegistry.GetDataInfo 3263
Registry.TRegistry.GetDataSize 3263
Registry.TRegistry.GetDataType 3264
Registry.TRegistry.GetKey 3264
Registry.TRegistry.GetKeyInfo 3264
Registry.TRegistry.GetKeyNames 3265
Registry.TRegistry.GetValueNames 3265
Registry.TRegistry.HasSubKeys 3266
Registry.TRegistry.KeyExists 3266
Registry.TRegistry.LazyWrite 3296
Registry.TRegistry.LoadKey 3268
Registry.TRegistry.MoveKey 3269
Registry.TRegistry.OpenKey 3270
Registry.TRegistry.OpenKeyReadOnly 3272
Registry.TRegistry.PutData 3273
Registry.TRegistry.ReadBinaryData 3273
Registry.TRegistry.ReadBool 3274
Registry.TRegistry.ReadCurrency 3274
Registry.TRegistry.ReadDate 3274
Registry.TRegistry.ReadDateTime 3275
Registry.TRegistry.ReadFloat 3275
Registry.TRegistry.ReadInteger 3276
Registry.TRegistry.ReadString 3276
Registry.TRegistry.ReadTime 3277
Registry.TRegistry.RegistryConnect 3277
Registry.TRegistry.RenameValue 3278
Registry.TRegistry.ReplaceKey 3278
Registry.TRegistry.RestoreKey 3279
Registry.TRegistry.RootKey 3297
Registry.TRegistry.SaveKey 3279
Registry.TRegistry.SetCurrentKey 3280
Registry.TRegistry.UnLoadKey 3280
Registry.TRegistry.ValueExists 3281
Registry.TRegistry.WriteBinaryData 3281
Registry.TRegistry.WriteBool 3282

Registry.TRegistry.WriteCurrency 3282
Registry.TRegistry.WriteDate 3282
Registry.TRegistry.WriteDateTime 3283
Registry.TRegistry.WriteExpandString 3286
Registry.TRegistry.WriteFloat 3288
Registry.TRegistry.WriteInteger 3289
Registry.TRegistry.WriteString 3289
Registry.TRegistry.WriteTime 3292
Registry.TRegistryIniFile 3299
Registry.TRegistryIniFile.Create 3304
Registry.TRegistryIniFile.DeleteKey 3305
Registry.TRegistryIniFile.Destroy 3304
Registry.TRegistryIniFile.EraseSection 3306
Registry.TRegistryIniFile.ReadBinaryStream 3306
Registry.TRegistryIniFile.ReadDate 3307
Registry.TRegistryIniFile.ReadDateTime 3307
Registry.TRegistryIniFile.ReadFloat 3308
Registry.TRegistryIniFile.ReadInteger 3308
Registry.TRegistryIniFile.ReadSection 3311
Registry.TRegistryIniFile.ReadSections 3312
Registry.TRegistryIniFile.ReadSectionValues 3313
Registry.TRegistryIniFile.ReadString 3313
Registry.TRegistryIniFile.ReadTime 3316
Registry.TRegistryIniFile.RegIniFile 3326
Registry.TRegistryIniFile.UpdateFile 3317
Registry.TRegistryIniFile.WriteBinaryStream 3317
Registry.TRegistryIniFile.WriteDate 3317
Registry.TRegistryIniFile.WriteDateTime 3318
Registry.TRegistryIniFile.WriteFloat 3319
Registry.TRegistryIniFile.WriteInteger 3319
Registry.TRegistryIniFile.WriteString 3322
Registry.TRegistryIniFile.WriteTime 3325
Registry.TRegKeyInfo 3328
Registry.TRegKeyInfo record 3328
RegistryConnect
TRegistry class 3277
reGroup enumeration member 4029
Release
TAutoDispatch class 2503
TAutoObject class 2510
TIBCustomSQLMonitor class 562
ReleaseMonitor
IIBSQLMonitorHook interface 570
ReleaseSID
ISecurityProperty interface 2427
reMenu enumeration member 4029
RemoteDatabasePath
TLimboTransactionInfo class 472
RemoteMachineName
TOleServer class 2719
RemoteSite
TLimboTransactionInfo class 472
Remove
TMenuItem class 1388
TOutlineNode class 2950
TPNGPointerList class 3001
TPopupList class 1429
TQuickAccessToolbarGroups class 3479
TStringHash class 785
RemoveAllTabs
TCustomRibbon class 3387
RemoveChunk
TPNGList class 2999
RemoveContainer
IVCLFrameForm interface 2616
RemoveDatabase
TIBTransaction class 213
RemoveDatabases
TIBTransaction class 213
RemoveEventNotifier
TIBDataBase class 187
RemoveLicense
TIBLicensingService class 435
RemoveMenus
TOleControl class 2675
RemoveObjectAsWriting
TSOAPDomConv class 2777
RemoveObjectToDestroy

TDataContext class 822
RemoveOnDataChange
IInetXWebDataSet interface 2217
RemoveRecentItem
TCustomRibbon class 3388
RemoveTab
TCustomRibbon class 3388
RemoveTransaction
TIBDataBase class 187
RemoveTransactions
TIBDataBase class 188
Rename
TMemIniFile class 777
RenameValue
TRegistry class 3278
Replace
TCustomImageList class 672
Replace enumeration member 480
ReplaceIcon
TCustomImageList class 673
ReplaceKey
TRegistry class 3278
ReplaceMasked
TCustomImageList class 673
RepositoryID
IStub interface 2814
ReqFiles 3331
ReqFiles namespace 3331
Classes 3332
ReqFiles.TWebRequestFile 3332
ReqFiles.TWebRequestFile.Create 3333
ReqFiles.TWebRequestFile.Destroy 3334
ReqFiles.TWebRequestFile.GetContentType 3334
ReqFiles.TWebRequestFile.GetFieldName 3334
ReqFiles.TWebRequestFile.GetFileName 3335
ReqFiles.TWebRequestFile.GetStream 3335
ReqFiles.TWebRequestFiles 3336
ReqFiles.TWebRequestFiles.Add 3338
ReqFiles.TWebRequestFiles.Create 3336
ReqFiles.TWebRequestFiles.Destroy 3337
ReqFiles.TWebRequestFiles.GetCount 3338
ReqFiles.TWebRequestFiles.GetItem 3339
ReqFiles.TWebRequestFileStream 3339
ReqFiles.TWebRequestFileStream.Create 3340
ReqFiles.TWebRequestFileStream.Write 3340
ReqMulti 3341
ReqMulti namespace 3341
Classes 3342
ReqMulti.TMultipartContentParser 3342
ReqMulti.TMultipartContentParser.BufferContent 3344
ReqMulti.TMultipartContentParser.CanParse 3344
ReqMulti.TMultipartContentParser.Destroy 3343
ReqMulti.TMultipartContentParser.ExtractContentTypeFields 3344
ReqMulti.TMultipartContentParser.ExtractMultipartContentFields 3345
ReqMulti.TMultipartContentParser.GetContentFields 3345
ReqMulti.TMultipartContentParser.GetFiles 3345
ReqMulti.TMultipartContentParser.ParseMultiPart 3346
ReqMulti.TMultipartContentParser.ParseMultiPartContent 3346
ReqMulti.TMultipartContentParser.ParseMultipartHeaders 3346
ReQuery
TIBCustomDataSet class 89
RequestAlign
TCustomRibbonPage class 3434
TCustomRibbonQuickAccessToolbar class 3450
RequestBorderSpace
TOleControl class 2675
RequestNewObjectLayout
TOleControl class 2676
reQuickAccessToolbar enumeration member 4029
RequiredWidth
TCustomRibbonGroup class 3424
reRadioButton enumeration member 4029
reRibbon enumeration member 4029
reScrollBar enumeration member 4029
Reset
IEnumNames interface 2418

TCustomMaskEdit class 1089
TDataPacketWriter class 3143
TDataSetProvider class 3157
ResetMultiRef
TSOAPDomConv class 2777
ResInstLoad
TCustomImageList class 673
Resize
TCustomRibbon class 3388
TCustomRibbonGroup class 3418
TCustomRibbonPage class 3434
Resolver
TBaseProvider class 3088
TUpdateTree class 3212
ResolveToDataSet
TDataSetProvider class 3162
ResourceLoad
TCustomImageList class 674
reSpinner enumeration member 4029
reStatusBar enumeration member 4029
RestoreDefaults
IRestoreDefaults interface 2222
TWebDataInput class 2092
RestoreFieldDefaults
TWebDataInput class 2092
TWebRadioGroupInput class 2126
TWebTextAreaInput class 2143
TWebTextInput class 2149
RestoreKey
TRegistry class 3279
Resume
TMediaPlayer class 2327
reTab enumeration member 4029
RethinkHotkeys
TMenuItem class 1388
RethinkLines
TMenuItem class 1389
RevertRecord
TIBCustomDataSet class 89
Rewind
TMediaPlayer class 2328
rgvCollapsed enumeration member 3572
rgvLarge enumeration member 3572
rgvMedium enumeration member 3572
rgvSmall enumeration member 3572
Ribbon 3347
IRibbon interface 3567
TCustomRibbonPage class 3438
TCustomRibbonQuickAccessToolbar class 3452
TRibbonTabs class 3565
Ribbon namespace 3347
Classes 3349
Interfaces 3566
Structs, Records, Enums 3571
Types 3576
Ribbon.ERibbonException 3349
Ribbon.IRibbon 3566
Ribbon.IRibbon.GetRibbon 3567
Ribbon.IRibbon.Ribbon 3567
Ribbon.IRibbon.SetRibbon 3567
Ribbon.IRibbonGroup 3568
Ribbon.IRibbonGroup.AlignGroupControls 3568
Ribbon.IRibbonGroup.CreateControl 3569
Ribbon.IRibbonGroup.StyleChanged 3569
Ribbon.IRibbonQuickAccessToolbar 3569
Ribbon.IRibbonQuickAccessToolbar.DoUpdateQATWidth 3570
Ribbon.TApplicationMenu 3350
Ribbon.TApplicationMenu.Assign 3352
Ribbon.TApplicationMenu.Caption 3353
Ribbon.TApplicationMenu.CommandType 3354
Ribbon.TApplicationMenu.Create 3351
Ribbon.TApplicationMenu.Destroy 3351
Ribbon.TApplicationMenu.GetActionClientItem 3352
Ribbon.TApplicationMenu.Icon 3354
Ribbon.TApplicationMenu.IconSize 3354
Ribbon.TApplicationMenu.Initialize 3353
Ribbon.TApplicationMenu.KeyTip 3355
Ribbon.TApplicationMenu.Menu 3355

Ribbon.TApplicationMenu.ScreenTip 3355
Ribbon.TApplicationMenu.Show 3353
Ribbon.TCommandType 3571
Ribbon.TCommandType enumeration 3571
Ribbon.TCustomActionControlBar 3355
Ribbon.TCustomActionControlBar.CMControlChange 3357
Ribbon.TCustomActionControlBar.Create 3356
Ribbon.TCustomActionControlBar.CreateControl 3357
Ribbon.TCustomActionControlBar.CreateControls 3357
Ribbon.TCustomActionControlBar.CreatingControls 3358
Ribbon.TCustomActionControlBar.GetControlClass 3357
Ribbon.TCustomRibbon 3358
Ribbon.TCustomRibbon.ActionManager 3394
Ribbon.TCustomRibbon.Active 3394
Ribbon.TCustomRibbon.ActivePage 3394
Ribbon.TCustomRibbon.AddRecentItem 3370
Ribbon.TCustomRibbon.AddTab 3370
Ribbon.TCustomRibbon.AllowKeyboardNavigation 3395
Ribbon.TCustomRibbon.ApplicationMenu 3395
Ribbon.TCustomRibbon.BeginUpdate 3370
Ribbon.TCustomRibbon.CheckRibbonSize 3370
Ribbon.TCustomRibbon.ClearRecentItems 3371
Ribbon.TCustomRibbon.CMBidimodechanged 3371
Ribbon.TCustomRibbon.CMDesignHitTest 3371
Ribbon.TCustomRibbon.CMFontchanged 3371
Ribbon.TCustomRibbon.CMMouseenter 3371
Ribbon.TCustomRibbon.CMMouseleave 3372
Ribbon.TCustomRibbon.CMTextchanged 3372
Ribbon.TCustomRibbon.CMThemeChanged 3372
Ribbon.TCustomRibbon.ColorMap 3395
Ribbon.TCustomRibbon.ContextMenuPopupHandler 3372
Ribbon.TCustomRibbon.Create 3362
Ribbon.TCustomRibbon.CreateContextMenu 3373
Ribbon.TCustomRibbon.cRibbonFirstTabOffSet 3364
Ribbon.TCustomRibbon.cRibbonGroupCaptionHeight 3364
Ribbon.TCustomRibbon.cRibbonGroupHeight 3364
Ribbon.TCustomRibbon.cRibbonHeight 3364
Ribbon.TCustomRibbon.cRibbonHideHeight 3365
Ribbon.TCustomRibbon.cRibbonHideWidth 3365
Ribbon.TCustomRibbon.cRibbonMinimizedHeight 3365
Ribbon.TCustomRibbon.cRibbonMinimumCaptionWidth 3365
Ribbon.TCustomRibbon.cRibbonPageHeight 3366
Ribbon.TCustomRibbon.cRibbonQATHeight 3366
Ribbon.TCustomRibbon.cRibbonQuickAccessToolbarLeft 3366
Ribbon.TCustomRibbon.cRibbonQuickAccessToolbarTop 3366
Ribbon.TCustomRibbon.cRibbonTabHeight 3366
Ribbon.TCustomRibbon.cRibbonTabScrollButtonWidth 3367
Ribbon.TCustomRibbon.cRibbonTabSpacing 3367
Ribbon.TCustomRibbon.cRibbonUnthemedCaptionHeight 3367
Ribbon.TCustomRibbon.CustomFrameSupported 3373
Ribbon.TCustomRibbon.CustomFrameWndProc 3373
Ribbon.TCustomRibbon.DblClick 3373
Ribbon.TCustomRibbon.DefineProperties 3374
Ribbon.TCustomRibbon.DeleteTab 3374
Ribbon.TCustomRibbon.Destroy 3363
Ribbon.TCustomRibbon.DisplayKeyTips 3374
Ribbon.TCustomRibbon.DoAddToQuickAccessToolbar 3375
Ribbon.TCustomRibbon.DoChangeQuickAccessToolbarPosition 3375
Ribbon.TCustomRibbon.DoContextPopup 3375
Ribbon.TCustomRibbon.DocumentName 3396
Ribbon.TCustomRibbon.DoCustomizeQuickAccessToolbar 3375
Ribbon.TCustomRibbon.DoGroupStyleChanged 3376
Ribbon.TCustomRibbon.DoHelpButtonClick 3376
Ribbon.TCustomRibbon.DoMinimizeRibbon 3376
Ribbon.TCustomRibbon.DoMouseWheelDown 3376
Ribbon.TCustomRibbon.DoMouseWheelEx 3377
Ribbon.TCustomRibbon.DoMouseWheelUp 3377
Ribbon.TCustomRibbon.DoRecentItemClick 3377
Ribbon.TCustomRibbon.DoScrollTabsLeft 3378
Ribbon.TCustomRibbon.DoScrollTabsRight 3378
Ribbon.TCustomRibbon.DoTabChange 3378
Ribbon.TCustomRibbon.DrawCaption 3378
Ribbon.TCustomRibbon.DrawQuickAccessToolbar 3379
Ribbon.TCustomRibbon.DrawSystemButtons 3379
Ribbon.TCustomRibbon.DrawTab 3379
Ribbon.TCustomRibbon.EndUpdate 3379
Ribbon.TCustomRibbon.GetContextMenuOptions 3379

Ribbon.TCustomRibbon.GetHelpButtonRect 3380
Ribbon.TCustomRibbon.GetIcon 3380
Ribbon.TCustomRibbon.GetRibbonHeight 3380
Ribbon.TCustomRibbon.GetRibbonMetric 3380
Ribbon.TCustomRibbon.GetTabAt 3381
Ribbon.TCustomRibbon.GetTabRect 3381
Ribbon.TCustomRibbon.GetTabWidth 3381
Ribbon.TCustomRibbon.HasApplicationMenu 3381
Ribbon.TCustomRibbon.HasQuickAccessToolbar 3382
Ribbon.TCustomRibbon.HideRibbon 3396
Ribbon.TCustomRibbon.HideTabs 3396
Ribbon.TCustomRibbon.HotTabIndex 3396
Ribbon.TCustomRibbon.Idle 3382
Ribbon.TCustomRibbon.InvalidateHeaderArea 3382
Ribbon.TCustomRibbon.InvalidateScrollButtons 3382
Ribbon.TCustomRibbon.InvalidateSystemMenuArea 3383
Ribbon.TCustomRibbon.InvalidateTabArea 3383
Ribbon.TCustomRibbon.KeyDown 3383
Ribbon.TCustomRibbon.KeyTipsVisible 3397
Ribbon.TCustomRibbon.Loaded 3384
Ribbon.TCustomRibbon.Menu 3397
Ribbon.TCustomRibbon.Minimized 3397
Ribbon.TCustomRibbon.MouseDown 3384
Ribbon.TCustomRibbon.MouseInSystemMenu 3385
Ribbon.TCustomRibbon.MouseMove 3385
Ribbon.TCustomRibbon.MouseUp 3385
Ribbon.TCustomRibbon.MoveGroup 3386
Ribbon.TCustomRibbon.Notification 3386
Ribbon.TCustomRibbon.OnHelpButtonClick 3400
Ribbon.TCustomRibbon.OnLeftTabScrollButtonClick 3400
Ribbon.TCustomRibbon.OnRecentItemClick 3401
Ribbon.TCustomRibbon.OnRightTabScrollButtonClick 3401
Ribbon.TCustomRibbon.OnTabChange 3401
Ribbon.TCustomRibbon.Paint 3387
Ribbon.TCustomRibbon.QuickAccessToolbar 3397
Ribbon.TCustomRibbon.RebuildRibbonMetrics 3387
Ribbon.TCustomRibbon.RemoveAllTabs 3387
Ribbon.TCustomRibbon.RemoveRecentItem 3388
Ribbon.TCustomRibbon.RemoveTab 3388
Ribbon.TCustomRibbon.Resize 3388
Ribbon.TCustomRibbon.ScaleFactor 3398
Ribbon.TCustomRibbon.ScreenTips 3398
Ribbon.TCustomRibbon.SetBounds 3389
Ribbon.TCustomRibbon.SetParent 3389
Ribbon.TCustomRibbon.ShowHelpButton 3398
Ribbon.TCustomRibbon.Style 3398
Ribbon.TCustomRibbon.TabHandledKeyTips 3390
Ribbon.TCustomRibbon.TabHideKeyTips 3390
Ribbon.TCustomRibbon.TabIndex 3399
Ribbon.TCustomRibbon.Tabs 3399
Ribbon.TCustomRibbon.TabShowKeyTips 3390
Ribbon.TCustomRibbon.TTabChangeEvent 3400
Ribbon.TCustomRibbon.UM_CHECKSIZE 3367
Ribbon.TCustomRibbon.UM_DISPLAYKEYTIPS 3368
Ribbon.TCustomRibbon.UMCheckResize 3391
Ribbon.TCustomRibbon.UMDisplayKeyTips 3391
Ribbon.TCustomRibbon.UpdateQATPosition 3391
Ribbon.TCustomRibbon.UpdateSystemMenuInfo 3391
Ribbon.TCustomRibbon.UseCustomFrame 3399
Ribbon.TCustomRibbon.ValidTab 3392
Ribbon.TCustomRibbon.WMEraseBkgnd 3392
Ribbon.TCustomRibbon.WMGetDlgCode 3392
Ribbon.TCustomRibbon.WMKillFocus 3392
Ribbon.TCustomRibbon.WMSettingChange 3393
Ribbon.TCustomRibbon.WMWindowPosChanged 3393
Ribbon.TCustomRibbonClass 3576
Ribbon.TCustomRibbonClass type 3576
Ribbon.TCustomRibbonColorMap 3401
Ribbon.TCustomRibbonColorMap.ActiveTabFontColor 3402
Ribbon.TCustomRibbonColorMap.BevelMinimizedInner 3403
Ribbon.TCustomRibbonColorMap.BevelMinimizedOuter 3403
Ribbon.TCustomRibbonColorMap.CaptionFontColor 3403
Ribbon.TCustomRibbonColorMap.ControlColor 3403
Ribbon.TCustomRibbonColorMap.ControlFrameColor 3404
Ribbon.TCustomRibbonColorMap.ControlInactiveColor 3404
Ribbon.TCustomRibbonColorMap.DocumentFontColor 3404
Ribbon.TCustomRibbonColorMap.GroupFontColor 3404
Ribbon.TCustomRibbonColorMap.InactiveTabFontColor 3404

Ribbon.TCustomRibbonColorMap.QuickAccessToolbarMoreBorderColor 3405
Ribbon.TCustomRibbonColorMap.QuickAccessToolbarMoreColor 3405
Ribbon.TCustomRibbonGroup 3405
Ribbon.TCustomRibbonGroup.AlignControls 3409
Ribbon.TCustomRibbonGroup.AlignGroupControls 3410
Ribbon.TCustomRibbonGroup.Assign 3410
Ribbon.TCustomRibbonGroup.AutoSizingChanged 3410
Ribbon.TCustomRibbonGroup.BarHandleKeyTips 3410
Ribbon.TCustomRibbonGroup.BarHideKeyTips 3411
Ribbon.TCustomRibbonGroup.BarShowKeyTips 3411
Ribbon.TCustomRibbonGroup.CanCollapse 3421
Ribbon.TCustomRibbonGroup.Clone 3411
Ribbon.TCustomRibbonGroup.CMFontchanged 3412
Ribbon.TCustomRibbonGroup.CMItemDropPoint 3412
Ribbon.TCustomRibbonGroup.CMMouseenter 3412
Ribbon.TCustomRibbonGroup.CMMouseleave 3412
Ribbon.TCustomRibbonGroup.CMTextchanged 3413
Ribbon.TCustomRibbonGroup.CollapsedControl 3421
Ribbon.TCustomRibbonGroup.Create 3407
Ribbon.TCustomRibbonGroup.CreateControl 3413
Ribbon.TCustomRibbonGroup.CreateControls 3413
Ribbon.TCustomRibbonGroup.Destroy 3408
Ribbon.TCustomRibbonGroup.DialogAction 3421
Ribbon.TCustomRibbonGroup.DockDrop 3413
Ribbon.TCustomRibbonGroup.DoDropItem 3414
Ribbon.TCustomRibbonGroup.DoEndDrag 3414
Ribbon.TCustomRibbonGroup.DragDrop 3414
Ribbon.TCustomRibbonGroup.DrawShowDialogButton 3414
Ribbon.TCustomRibbonGroup.GetCaptionHeight 3414
Ribbon.TCustomRibbonGroup.GetGroupHeight 3415
Ribbon.TCustomRibbonGroup.GetKeyTip 3415
Ribbon.TCustomRibbonGroup.GroupAlign 3421
Ribbon.TCustomRibbonGroup.GroupIndex 3422
Ribbon.TCustomRibbonGroup.GroupVariant 3422
Ribbon.TCustomRibbonGroup.GroupWidth 3422
Ribbon.TCustomRibbonGroup.HandleKeyTip 3415
Ribbon.TCustomRibbonGroup.HasGallery 3423
Ribbon.TCustomRibbonGroup.HideControls 3416
Ribbon.TCustomRibbonGroup.HideKeyTip 3416
Ribbon.TCustomRibbonGroup.Items 3423
Ribbon.TCustomRibbonGroup.KeyTip 3423
Ribbon.TCustomRibbonGroup.KeyTipClick 3416
Ribbon.TCustomRibbonGroup.MouseDown 3417
Ribbon.TCustomRibbonGroup.MouseMove 3417
Ribbon.TCustomRibbonGroup.MouseUp 3417
Ribbon.TCustomRibbonGroup.Notification 3417
Ribbon.TCustomRibbonGroup.Page 3423
Ribbon.TCustomRibbonGroup.Paint 3418
Ribbon.TCustomRibbonGroup.RequiredWidth 3424
Ribbon.TCustomRibbonGroup.Resize 3418
Ribbon.TCustomRibbonGroup.Rows 3424
Ribbon.TCustomRibbonGroup.SetAutoSizing 3418
Ribbon.TCustomRibbonGroup.SetBounds 3418
Ribbon.TCustomRibbonGroup.SetKeyTip 3418
Ribbon.TCustomRibbonGroup.SetParent 3419
Ribbon.TCustomRibbonGroup.SetupDropDownBtn 3419
Ribbon.TCustomRibbonGroup.ShowControls 3419
Ribbon.TCustomRibbonGroup.ShowKeyTip 3419
Ribbon.TCustomRibbonGroup.StyleChanged 3420
Ribbon.TCustomRibbonGroup.TGroupAlign 3408
Ribbon.TCustomRibbonGroup.TGroupAlign enumeration 3408
Ribbon.TCustomRibbonGroup.TRowPosition 3424
Ribbon.TCustomRibbonPage 3425
Ribbon.TCustomRibbonPage.AddGroup 3429
Ribbon.TCustomRibbonPage.AlignGroups 3429
Ribbon.TCustomRibbonPage.CMControlListChanging 3429
Ribbon.TCustomRibbonPage.CMTextchanged 3429
Ribbon.TCustomRibbonPage.CollapseGroup 3430
Ribbon.TCustomRibbonPage.CollapseGroupCount 3437
Ribbon.TCustomRibbonPage.Create 3426
Ribbon.TCustomRibbonPage.CreateScrollButton 3430
Ribbon.TCustomRibbonPage.cRibbonGroupLeftOffset 3428
Ribbon.TCustomRibbonPage.cRibbonGroupSpacing 3428
Ribbon.TCustomRibbonPage.Destroy 3427
Ribbon.TCustomRibbonPage.DoScrollButtonClick 3430
Ribbon.TCustomRibbonPage.FindGroup 3430

Ribbon.TCustomRibbonPage.FindGroupAt 3430
Ribbon.TCustomRibbonPage.GetGroupPos 3431
Ribbon.TCustomRibbonPage.GetNextVisibleGroup 3431
Ribbon.TCustomRibbonPage.GetPageHeight 3431
Ribbon.TCustomRibbonPage.GetPageTop 3431
Ribbon.TCustomRibbonPage.GroupCount 3437
Ribbon.TCustomRibbonPage.GroupOffset 3437
Ribbon.TCustomRibbonPage.Groups 3437
Ribbon.TCustomRibbonPage.HandleKeyTips 3432
Ribbon.TCustomRibbonPage.HideKeyTips 3432
Ribbon.TCustomRibbonPage.Index 3438
Ribbon.TCustomRibbonPage.Invalidate 3432
Ribbon.TCustomRibbonPage.KeyTip 3438
Ribbon.TCustomRibbonPage.Loaded 3432
Ribbon.TCustomRibbonPage.Notification 3433
Ribbon.TCustomRibbonPage.Paint 3433
Ribbon.TCustomRibbonPage.RequestAlign 3434
Ribbon.TCustomRibbonPage.Resize 3434
Ribbon.TCustomRibbonPage.Ribbon 3438
Ribbon.TCustomRibbonPage.RibbonStyle 3434
Ribbon.TCustomRibbonPage.SetBounds 3435
Ribbon.TCustomRibbonPage.SetParent 3435
Ribbon.TCustomRibbonPage.ShowKeyTips 3436
Ribbon.TCustomRibbonPage.UpdateActionManager 3436
Ribbon.TCustomRibbonPage.UpdateScrollButtons 3436
Ribbon.TCustomRibbonQuickAccessPopupToolbar 3438
Ribbon.TCustomRibbonQuickAccessPopupToolbar.Create 3439
Ribbon.TCustomRibbonQuickAccessPopupToolbar.CreateControl 3440
Ribbon.TCustomRibbonQuickAccessPopupToolbar.CreateWnd 3440
Ribbon.TCustomRibbonQuickAccessPopupToolbar.DrawBackground 3440
Ribbon.TCustomRibbonQuickAccessPopupToolbar.GetControlClass 3441
Ribbon.TCustomRibbonQuickAccessPopupToolbar.NCPaint 3441
Ribbon.TCustomRibbonQuickAccessPopupToolbar.ValidOrientations 3441
Ribbon.TCustomRibbonQuickAccessToolbar 3441
Ribbon.TCustomRibbonQuickAccessToolbar.BarHandleKeyTips 3444
Ribbon.TCustomRibbonQuickAccessToolbar.BarHideKeyTips 3444
Ribbon.TCustomRibbonQuickAccessToolbar.BarShowKeyTips 3445
Ribbon.TCustomRibbonQuickAccessToolbar.CMFontchanged 3445
Ribbon.TCustomRibbonQuickAccessToolbar.Create 3443
Ribbon.TCustomRibbonQuickAccessToolbar.CreateControl 3445
Ribbon.TCustomRibbonQuickAccessToolbar.CreateControls 3446
Ribbon.TCustomRibbonQuickAccessToolbar.Customize 3446
Ribbon.TCustomRibbonQuickAccessToolbar.DoContextPopup 3446
Ribbon.TCustomRibbonQuickAccessToolbar.DoDropActions 3446
Ribbon.TCustomRibbonQuickAccessToolbar.DoDropCategory 3447
Ribbon.TCustomRibbonQuickAccessToolbar.DoDropItem 3447
Ribbon.TCustomRibbonQuickAccessToolbar.DoMoreCommands 3447
Ribbon.TCustomRibbonQuickAccessToolbar.DoScrollBtnClick 3447
Ribbon.TCustomRibbonQuickAccessToolbar.DoUpdateQATWidth 3448
Ribbon.TCustomRibbonQuickAccessToolbar.DragDrop 3448
Ribbon.TCustomRibbonQuickAccessToolbar.DrawBackground 3448
Ribbon.TCustomRibbonQuickAccessToolbar.GetBarHeight 3448
Ribbon.TCustomRibbonQuickAccessToolbar.GetBarWidth 3449
Ribbon.TCustomRibbonQuickAccessToolbar.GetScrollBtnClass 3449
Ribbon.TCustomRibbonQuickAccessToolbar.Items 3452
Ribbon.TCustomRibbonQuickAccessToolbar.Notification 3449
Ribbon.TCustomRibbonQuickAccessToolbar.OnMoreCommands 3452
Ribbon.TCustomRibbonQuickAccessToolbar.RequestAlign 3450
Ribbon.TCustomRibbonQuickAccessToolbar.Ribbon 3452

Ribbon.TCustomRibbonQuickAccessToolbar.ScrollMode 3453
Ribbon.TCustomRibbonQuickAccessToolbar.SetParent 3450
Ribbon.TCustomRibbonQuickAccessToolbar.SetupDropDownBtn 3451
Ribbon.TCustomRibbonQuickAccessToolbar.ShowBorder 3453
Ribbon.TCustomRibbonQuickAccessToolbar.ShowCustomizeMenu 3451
Ribbon.TCustomRibbonQuickAccessToolbar.ShowMoreControls 3451
Ribbon.TCustomRibbonQuickAccessToolbar.TMoreCommandsEvent 3453
Ribbon.TCustomRibbonQuickAccessToolbar.UpdateQATWidth 3451
Ribbon.TCustomTabData 3453
Ribbon.TCustomTabData.Add 3455
Ribbon.TCustomTabData.AvailableSpace 3456
Ribbon.TCustomTabData.CalculateTabWidths 3455
Ribbon.TCustomTabData.Clear 3456
Ribbon.TCustomTabData.Create 3454
Ribbon.TCustomTabData.Destroy 3455
Ribbon.TCustomTabData.FirstVisibleTabIndex 3456
Ribbon.TCustomTabData.FullWidth 3457
Ribbon.TCustomTabData.ItemCount 3457
Ribbon.TCustomTabData.LastVisibleTabIndex 3457
Ribbon.TCustomTabData.LeftTabScrollButtonRect 3457
Ribbon.TCustomTabData.NewWidth 3458
Ribbon.TCustomTabData.Owner 3458
Ribbon.TCustomTabData.RightTabScrollButtonRect 3458
Ribbon.TCustomTabData.ScrollButtons 3458
Ribbon.TCustomTabData.SeparatorOpacity 3458
Ribbon.TCustomTabData.TabSeparatorState 3459
Ribbon.TCustomTabData.TabUpdateRect 3459
Ribbon.THelpButton 3459
Ribbon.THelpButton.DrawImage 3460
Ribbon.TIconSize 3572
Ribbon.TIconSize enumeration 3572
Ribbon.TOffice2007Button 3461
Ribbon.TOffice2007Button.CMMouseenter 3463
Ribbon.TOffice2007Button.CMMouseleave 3463
Ribbon.TOffice2007Button.Create 3462
Ribbon.TOffice2007Button.Destroy 3462
Ribbon.TOffice2007Button.DoClick 3463
Ribbon.TOffice2007Button.DrawBackground 3464
Ribbon.TOffice2007Button.DrawImage 3464
Ribbon.TOffice2007Button.MouseDown 3464
Ribbon.TOffice2007Button.MouseInControl 3467
Ribbon.TOffice2007Button.MouseMove 3465
Ribbon.TOffice2007Button.MouseUp 3465
Ribbon.TOffice2007Button.OnClick 3467
Ribbon.TOffice2007Button.Paint 3466
Ribbon.TOffice2007Button.RibbonStyle 3467
Ribbon.TOffice2007Button.SetParent 3466
Ribbon.TQuickAccessToolbar 3469
Ribbon.TQuickAccessToolbar.ActionBar 3473
Ribbon.TQuickAccessToolbar.Assign 3472
Ribbon.TQuickAccessToolbar.BarWidth 3474
Ribbon.TQuickAccessToolbar.Create 3470
Ribbon.TQuickAccessToolbar.DoChange 3472
Ribbon.TQuickAccessToolbar.OnChange 3475
Ribbon.TQuickAccessToolbar.Position 3474
Ribbon.TQuickAccessToolbar.Show 3473
Ribbon.TQuickAccessToolbar.ShowMinimize 3474
Ribbon.TQuickAccessToolbar.ShowMoreCommands 3474
Ribbon.TQuickAccessToolbar.ShowPosition 3474
Ribbon.TQuickAccessToolbar.TogglePosition 3473
Ribbon.TQuickAccessToolbar.TQATPosition 3471
Ribbon.TQuickAccessToolbar.TQATPosition enumeration 3471
Ribbon.TQuickAccessToolbarGroupItem 3475
Ribbon.TQuickAccessToolbarGroupItem.Action 3476
Ribbon.TQuickAccessToolbarGroupItem.Caption 3477
Ribbon.TQuickAccessToolbarGroupItem.Create 3476
Ribbon.TQuickAccessToolbarGroupItem.Destroy 3476
Ribbon.TQuickAccessToolbarGroupItem.Group 3477
Ribbon.TQuickAccessToolbarGroups 3477
Ribbon.TQuickAccessToolbarGroups.Add 3478
Ribbon.TQuickAccessToolbarGroups.Clean 3479
Ribbon.TQuickAccessToolbarGroups.Create 3478

Ribbon.TQuickAccessToolbarGroups.Destroy 3478
Ribbon.TQuickAccessToolbarGroups.Exists 3479
Ribbon.TQuickAccessToolbarGroups.Group 3479
Ribbon.TQuickAccessToolbarGroups.Remove 3479
Ribbon.TRecentItemClickEvent 3576
Ribbon.TRecentItemClickEvent type 3576
Ribbon.TRibbon 3480
Ribbon.TRibbon.ActionManager 3484
Ribbon.TRibbon.Align 3484
Ribbon.TRibbon.Anchors 3488
Ribbon.TRibbon.ApplicationMenu 3488
Ribbon.TRibbon.BiDiMode 3488
Ribbon.TRibbon.Caption 3489
Ribbon.TRibbon.DocumentName 3492
Ribbon.TRibbon.Enabled 3492
Ribbon.TRibbon.Font 3492
Ribbon.TRibbon.Height 3493
Ribbon.TRibbon.HideTabs 3495
Ribbon.TRibbon.OnHelpButtonClick 3498
Ribbon.TRibbon.OnRecentItemClick 3498
Ribbon.TRibbon.OnTabChange 3498
Ribbon.TRibbon.ParentBiDiMode 3495
Ribbon.TRibbon.ParentFont 3495
Ribbon.TRibbon.QuickAccessToolbar 3496
Ribbon.TRibbon.ScreenTips 3496
Ribbon.TRibbon.ShowHelpButton 3496
Ribbon.TRibbon.Style 3497
Ribbon.TRibbon.TabIndex 3497
Ribbon.TRibbon.Tabs 3497
Ribbon.TRibbon.UseCustomFrame 3497
Ribbon.TRibbonCustomizeDlg 3499
Ribbon.TRibbonCustomizeDlg.SetupDlg 3499
Ribbon.TRibbonGroup 3500
Ribbon.TRibbonGroup.ActionManager 3503
Ribbon.TRibbonGroup.BiDiMode 3503
Ribbon.TRibbonGroup.Caption 3503
Ribbon.TRibbonGroup.DialogAction 3504
Ribbon.TRibbonGroup.GroupAlign 3504
Ribbon.TRibbonGroup.GroupIndex 3504
Ribbon.TRibbonGroup.Height 3505
Ribbon.TRibbonGroup.KeyTip 3505
Ribbon.TRibbonGroup.ParentBiDiMode 3505
Ribbon.TRibbonGroup.Rows 3505
Ribbon.TRibbonGroupCollapsedControl 3506
Ribbon.TRibbonGroupCollapsedControl.ActionManager 3515
Ribbon.TRibbonGroupCollapsedControl.AdjustBounds 3509
Ribbon.TRibbonGroupCollapsedControl.Caption 3515
Ribbon.TRibbonGroupCollapsedControl.Click 3509
Ribbon.TRibbonGroupCollapsedControl.CloseUp 3509
Ribbon.TRibbonGroupCollapsedControl.CMFontchanged 3510
Ribbon.TRibbonGroupCollapsedControl.CMMouseenter 3510
Ribbon.TRibbonGroupCollapsedControl.CMMouseleave 3510
Ribbon.TRibbonGroupCollapsedControl.CMTextchanged 3510
Ribbon.TRibbonGroupCollapsedControl.Create 3507
Ribbon.TRibbonGroupCollapsedControl.Destroy 3508
Ribbon.TRibbonGroupCollapsedControl.GetKeyTip 3510
Ribbon.TRibbonGroupCollapsedControl.GetParentControl 3511
Ribbon.TRibbonGroupCollapsedControl.GetTopPosition 3511
Ribbon.TRibbonGroupCollapsedControl.Group 3518
Ribbon.TRibbonGroupCollapsedControl.HandleKeyTip 3511
Ribbon.TRibbonGroupCollapsedControl.HideKeyTip 3512
Ribbon.TRibbonGroupCollapsedControl.KeyTip 3518
Ribbon.TRibbonGroupCollapsedControl.KeyTipClick 3512
Ribbon.TRibbonGroupCollapsedControl.MouseInControl 3518
Ribbon.TRibbonGroupCollapsedControl.Notification 3512
Ribbon.TRibbonGroupCollapsedControl.OldWidth 3518
Ribbon.TRibbonGroupCollapsedControl.Paint 3513
Ribbon.TRibbonGroupCollapsedControl.SetKeyTip 3513
Ribbon.TRibbonGroupCollapsedControl.SetupGroupImage 3514
Ribbon.TRibbonGroupCollapsedControl.ShowKeyTip 3514
Ribbon.TRibbonGroupDockForm 3519
Ribbon.TRibbonGroupDockForm.Control 3521
Ribbon.TRibbonGroupDockForm.Create 3519
Ribbon.TRibbonGroupDockForm.CreateParams 3520
Ribbon.TRibbonGroupDockForm.ParentControl 3521
Ribbon.TRibbonGroupDockForm.WMActivate 3520
Ribbon.TRibbonGroupScrollButton 3521
Ribbon.TRibbonGroupScrollButton.CMDesignHitTest 3523

Ribbon.TRibbonGroupScrollButton.CMMouseenter 3523
Ribbon.TRibbonGroupScrollButton.CMMouseleave 3523
Ribbon.TRibbonGroupScrollButton.Create 3522
Ribbon.TRibbonGroupScrollButton.Direction 3524
Ribbon.TRibbonGroupScrollButton.MouseInControl 3524
Ribbon.TRibbonGroupScrollButton.Paint 3523
Ribbon.TRibbonGroupScrollButton.RibbonStyle 3524
Ribbon.TRibbonGroupVariant 3572
Ribbon.TRibbonGroupVariant enumeration 3572
Ribbon.TRibbonLunaColorMap 3525
Ribbon.TRibbonLunaColorMap.ActiveTabFontColor 3527
Ribbon.TRibbonLunaColorMap.BevelMinimizedInner 3527
Ribbon.TRibbonLunaColorMap.BevelMinimizedOuter 3527
Ribbon.TRibbonLunaColorMap.BtnSelectedFont 3527
Ribbon.TRibbonLunaColorMap.CaptionFontColor 3528
Ribbon.TRibbonLunaColorMap.ControlColor 3528
Ribbon.TRibbonLunaColorMap.ControlFrameColor 3528
Ribbon.TRibbonLunaColorMap.ControlInactiveColor 3528
Ribbon.TRibbonLunaColorMap.DisabledFontColor 3528
Ribbon.TRibbonLunaColorMap.DocumentFontColor 3529
Ribbon.TRibbonLunaColorMap.FontColor 3529
Ribbon.TRibbonLunaColorMap.GroupFontColor 3529
Ribbon.TRibbonLunaColorMap.HighlightColor 3529
Ribbon.TRibbonLunaColorMap.InactiveTabFontColor 3530
Ribbon.TRibbonLunaColorMap.QuickAccessToolbarMoreBorderColor 3530
Ribbon.TRibbonLunaColorMap.QuickAccessToolbarMoreColor 3530
Ribbon.TRibbonLunaColorMap.UpdateColors 3526
Ribbon.TRibbonMetric 3573
Ribbon.TRibbonMetric enumeration 3573
Ribbon.TRibbonObsidianColorMap 3530
Ribbon.TRibbonObsidianColorMap.ActiveTabFontColor 3532
Ribbon.TRibbonObsidianColorMap.BevelMinimizedInner 3533
Ribbon.TRibbonObsidianColorMap.BevelMinimizedOuter 3533
Ribbon.TRibbonObsidianColorMap.BtnSelectedFont 3533
Ribbon.TRibbonObsidianColorMap.CaptionFontColor 3533
Ribbon.TRibbonObsidianColorMap.ControlColor 3533
Ribbon.TRibbonObsidianColorMap.ControlFrameColor 3534
Ribbon.TRibbonObsidianColorMap.ControlInactiveColor 3534
Ribbon.TRibbonObsidianColorMap.DisabledFontColor 3534
Ribbon.TRibbonObsidianColorMap.DocumentFontColor 3534
Ribbon.TRibbonObsidianColorMap.FontColor 3535
Ribbon.TRibbonObsidianColorMap.GroupFontColor 3535
Ribbon.TRibbonObsidianColorMap.HighlightColor 3535
Ribbon.TRibbonObsidianColorMap.InactiveTabFontColor 3535
Ribbon.TRibbonObsidianColorMap.QuickAccessToolbarMoreBorderColor 3536
Ribbon.TRibbonObsidianColorMap.QuickAccessToolbarMoreColor 3536
Ribbon.TRibbonObsidianColorMap.UpdateColors 3532
Ribbon.TRibbonPage 3536
Ribbon.TRibbonPage.BiDiMode 3538
Ribbon.TRibbonPage.Caption 3539
Ribbon.TRibbonPage.Height 3541
Ribbon.TRibbonPage.Index 3542
Ribbon.TRibbonPage.KeyTip 3543
Ribbon.TRibbonPage.Name 3543
Ribbon.TRibbonPage.ParentBiDiMode 3545
Ribbon.TRibbonQuickAccessPopupToolbar 3545
Ribbon.TRibbonQuickAccessToolbar 3546
Ribbon.TRibbonQuickAccessToolbar.ActionManager 3548
Ribbon.TRibbonQuickAccessToolbar.OnMoreCommands 3548
Ribbon.TRibbonQuickAccessToolbar.VertMargin 3548
Ribbon.TRibbonScrollButton 3573
Ribbon.TRibbonScrollButton enumeration 3573
Ribbon.TRibbonScrollButtons 3577
Ribbon.TRibbonScrollButtons type 3577
Ribbon.TRibbonSilverColorMap 3549
Ribbon.TRibbonSilverColorMap.ActiveTabFontColor 3551
Ribbon.TRibbonSilverColorMap.BevelMinimizedInner 3551
Ribbon.TRibbonSilverColorMap.BevelMinimizedOuter 3551
Ribbon.TRibbonSilverColorMap.BtnSelectedFont 3551
Ribbon.TRibbonSilverColorMap.CaptionFontColor 3552
Ribbon.TRibbonSilverColorMap.ControlColor 3552
Ribbon.TRibbonSilverColorMap.ControlFrameColor 3552
Ribbon.TRibbonSilverColorMap.ControlInactiveColor 3552
Ribbon.TRibbonSilverColorMap.DisabledFontColor 3553

Ribbon.TRibbonSilverColorMap.DocumentFontColor 3553
Ribbon.TRibbonSilverColorMap.FontColor 3553
Ribbon.TRibbonSilverColorMap.GroupFontColor 3553
Ribbon.TRibbonSilverColorMap.HighlightColor 3554
Ribbon.TRibbonSilverColorMap.InactiveTabFontColor 3554
Ribbon.TRibbonSilverColorMap.QuickAccessToolbarMoreBorderColor 3554
Ribbon.TRibbonSilverColorMap.QuickAccessToolbarMoreColor 3554
Ribbon.TRibbonSilverColorMap.UpdateColors 3550
Ribbon.TRibbonTabItem 3555
Ribbon.TRibbonTabItem.Assign 3556
Ribbon.TRibbonTabItem.BestTabWidth 3558
Ribbon.TRibbonTabItem.Caption 3558
Ribbon.TRibbonTabItem.Destroy 3555
Ribbon.TRibbonTabItem.GetDisplayName 3557
Ribbon.TRibbonTabItem.KeyTip 3558
Ribbon.TRibbonTabItem.KeyTipForm 3558
Ribbon.TRibbonTabItem.MinTabwidth 3558
Ribbon.TRibbonTabItem.MinTruncatedWidth 3559
Ribbon.TRibbonTabItem.Modified 3559
Ribbon.TRibbonTabItem.Page 3559
Ribbon.TRibbonTabItem.X 3559
Ribbon.TRibbonTabItem.Y 3560
Ribbon.TRibbonTabs 3560
Ribbon.TRibbonTabs.Add 3561
Ribbon.TRibbonTabs.Create 3561
Ribbon.TRibbonTabs.DesignerAdd 3564
Ribbon.TRibbonTabs.FindGroup 3562
Ribbon.TRibbonTabs.GetPageFromPageCaption 3562
Ribbon.TRibbonTabs.IndexOfCaption 3562
Ribbon.TRibbonTabs.Items 3564
Ribbon.TRibbonTabs.Notify 3562
Ribbon.TRibbonTabs.Ribbon 3565
Ribbon.TRibbonTabs.SelectNext 3563
Ribbon.TRibbonTabs.SelectPrior 3563
Ribbon.TRibbonTabs.TabExists 3564
Ribbon.TTabActiveScrollButton 3574
Ribbon.TTabActiveScrollButton enumeration 3574
Ribbon.TTabSeparatorState 3574
Ribbon.TTabSeparatorState enumeration 3574
RibbonActnCtrls 3579
RibbonActnCtrls namespace 3579
Classes 3582
Constants 3932
Functions 3924
Interfaces 3912
Structs, Records, Enums 3926
Types 3931
RibbonActnCtrls.DrawLine 3924
RibbonActnCtrls.DrawLine function 3924
RibbonActnCtrls.IRibbonBarKeyTip 3912
RibbonActnCtrls.IRibbonBarKeyTip.BarHandleKeyTips 3913
RibbonActnCtrls.IRibbonBarKeyTip.BarHideKeyTips 3913
RibbonActnCtrls.IRibbonBarKeyTip.BarShowKeyTips 3914
RibbonActnCtrls.IRibbonContainedControl 3914
RibbonActnCtrls.IRibbonContainedControl.ContainedControl 3915
RibbonActnCtrls.IRibbonContainedControl.GetContainedControl 3915
RibbonActnCtrls.IRibbonContainedControl.SetContainedControl 3915
RibbonActnCtrls.IRibbonControl 3916
RibbonActnCtrls.IRibbonControl.CanAddToQuickAccessToolbar 3916
RibbonActnCtrls.IRibbonGroupDropdown 3917
RibbonActnCtrls.IRibbonGroupDropdown.CloseUp 3917
RibbonActnCtrls.IRibbonGroupDropdown.GetParentControl 3918
RibbonActnCtrls.IRibbonGroupDropdown.GetTopPosition 3918
RibbonActnCtrls.IRibbonKeyTip 3918
RibbonActnCtrls.IRibbonKeyTip.GetKeyTip 3919
RibbonActnCtrls.IRibbonKeyTip.HandleKeyTip 3920
RibbonActnCtrls.IRibbonKeyTip.HideKeyTip 3920
RibbonActnCtrls.IRibbonKeyTip.KeyTip 3922
RibbonActnCtrls.IRibbonKeyTip.KeyTipClick 3920
RibbonActnCtrls.IRibbonKeyTip.SetKeyTip 3921
RibbonActnCtrls.IRibbonKeyTip.ShowKeyTip 3921
RibbonActnCtrls.IRibbonSplitButton 3922

RibbonActnCtrls.IRibbonSplitButton.UpdateSplitBounds 3923
RibbonActnCtrls.IsLeftMouseButtonPressed 3924
RibbonActnCtrls.IsLeftMouseButtonPressed function 3924
RibbonActnCtrls.ShowKeyTip 3924
RibbonActnCtrls.ShowKeyTip function 3924
RibbonActnCtrls.TCustomRibbonComboBox 3583
RibbonActnCtrls.TCustomRibbonComboBox.BeginAutoDrag 3585
RibbonActnCtrls.TCustomRibbonComboBox.CMMouseenter 3585
RibbonActnCtrls.TCustomRibbonComboBox.CMMouseleave 3586
RibbonActnCtrls.TCustomRibbonComboBox.Color 3591
RibbonActnCtrls.TCustomRibbonComboBox.Create 3584
RibbonActnCtrls.TCustomRibbonComboBox.Destroy 3584
RibbonActnCtrls.TCustomRibbonComboBox.DragDrop 3586
RibbonActnCtrls.TCustomRibbonComboBox.DragOver 3586
RibbonActnCtrls.TCustomRibbonComboBox.Items 3592
RibbonActnCtrls.TCustomRibbonComboBox.KeyDown 3587
RibbonActnCtrls.TCustomRibbonComboBox.KeyPress 3587
RibbonActnCtrls.TCustomRibbonComboBox.MouseInControl 3592
RibbonActnCtrls.TCustomRibbonComboBox.MouseMove 3588
RibbonActnCtrls.TCustomRibbonComboBox.Notification 3588
RibbonActnCtrls.TCustomRibbonComboBox.SetBounds 3589
RibbonActnCtrls.TCustomRibbonComboBox.SetParent 3589
RibbonActnCtrls.TCustomRibbonComboBox.WMKillFocus 3590
RibbonActnCtrls.TCustomRibbonComboBox.WMSetFocus 3590
RibbonActnCtrls.TKeyTipHandled 3926
RibbonActnCtrls.TKeyTipHandled enumeration 3926
RibbonActnCtrls.TKeyTipList 3592
RibbonActnCtrls.TKeyTipList.Add 3594
RibbonActnCtrls.TKeyTipList.Clear 3594
RibbonActnCtrls.TKeyTipList.Count 3594
RibbonActnCtrls.TKeyTipList.Create 3593
RibbonActnCtrls.TKeyTipList.Current 3595
RibbonActnCtrls.TKeyTipList.Destroy 3593
RibbonActnCtrls.TKeyTipList.Full 3595
RibbonActnCtrls.TKeyTipList.Items 3595
RibbonActnCtrls.TKeyTipList.Partial 3596
RibbonActnCtrls.TKeyTipListEnumerator 3596
RibbonActnCtrls.TKeyTipListEnumerator.Create 3597
RibbonActnCtrls.TKeyTipListEnumerator.Current 3598
RibbonActnCtrls.TKeyTipListEnumerator.GetCurrent 3597
RibbonActnCtrls.TKeyTipListEnumerator.MoveNext 3597
RibbonActnCtrls.TKeyTipMatchItem 3598
RibbonActnCtrls.TKeyTipMatchItem.Control 3599
RibbonActnCtrls.TKeyTipMatchItem.Create 3599
RibbonActnCtrls.TKeyTipMatchItem.Handled 3599
RibbonActnCtrls.TKeyTipMatchItem.Index 3600
RibbonActnCtrls.TKeyTipMatchItem.MatchType 3600
RibbonActnCtrls.TKeyTipMatchType 3927
RibbonActnCtrls.TKeyTipMatchType enumeration 3927
RibbonActnCtrls.TKeyTipPosition 3927
RibbonActnCtrls.TKeyTipPosition enumeration 3927
RibbonActnCtrls.TKeyTipState 3928
RibbonActnCtrls.TKeyTipState enumeration 3928
RibbonActnCtrls.TKeyTipValidator 3600
RibbonActnCtrls.TKeyTipValidator.Validate 3601
RibbonActnCtrls.TNumGlyphs 3931
RibbonActnCtrls.TNumGlyphs type 3931
RibbonActnCtrls.TQATScrollMode 3929
RibbonActnCtrls.TQATScrollMode enumeration 3929
RibbonActnCtrls.TRibbonAddRemoveItem 3601
RibbonActnCtrls.TRibbonAddRemoveItem.DrawBackground 3602
RibbonActnCtrls.TRibbonAddRemoveItem.DrawGlyph 3602
RibbonActnCtrls.TRibbonApplicationButton 3603
RibbonActnCtrls.TRibbonApplicationButton.CalcBounds 3603
RibbonActnCtrls.TRibbonApplicationMenuButton 3604
RibbonActnCtrls.TRibbonApplicationMenuButton.CalcBounds 3606
RibbonActnCtrls.TRibbonApplicationMenuButton.CMMouseEnter 3607
RibbonActnCtrls.TRibbonApplicationMenuButton.CMMouseLeave 3607
RibbonActnCtrls.TRibbonApplicationMenuButton.Create 3605
RibbonActnCtrls.TRibbonApplicationMenuButton.Destroy 3605
RibbonActnCtrls.TRibbonApplicationMenuButton.DragOver 3607
RibbonActnCtrls.TRibbonApplicationMenuButton.GetKeyTip

3607

RibbonActnCtrls.TRibbonApplicationMenuButton.HandleKeyTip 3608

RibbonActnCtrls.TRibbonApplicationMenuButton.HideKeyTip 3608

RibbonActnCtrls.TRibbonApplicationMenuButton.KeyTip 3612

RibbonActnCtrls.TRibbonApplicationMenuButton.KeyTipClick 3609

RibbonActnCtrls.TRibbonApplicationMenuButton.MouseInControl 3612

RibbonActnCtrls.TRibbonApplicationMenuButton.Paint 3609

RibbonActnCtrls.TRibbonApplicationMenuButton.RibbonStyle 3609

RibbonActnCtrls.TRibbonApplicationMenuButton.SetKeyTip 3610

RibbonActnCtrls.TRibbonApplicationMenuButton.SetParent 3610

RibbonActnCtrls.TRibbonApplicationMenuButton.ShowKeyTip 3611

RibbonActnCtrls.TRibbonApplicationMenuButton.UMFadeControl 3611

RibbonActnCtrls.TRibbonApplicationMenuButton.WMLButtonDblClk 3611

RibbonActnCtrls.TRibbonApplicationMenuItem 3612

RibbonActnCtrls.TRibbonApplicationMenuItem.CalcBounds 3614

RibbonActnCtrls.TRibbonApplicationMenuItem.Create 3613

RibbonActnCtrls.TRibbonApplicationMenuItem.DrawGlyph 3614

RibbonActnCtrls.TRibbonApplicationMenuItem.DrawSelectedBackground 3615

RibbonActnCtrls.TRibbonApplicationMenuItem.DrawSeparator 3615

RibbonActnCtrls.TRibbonApplicationMenuItem.DrawShadowedText 3615

RibbonActnCtrls.TRibbonApplicationMenuItem.DrawSubMenuGlyph 3615

RibbonActnCtrls.TRibbonApplicationMenuItem.DrawText 3616

RibbonActnCtrls.TRibbonApplicationMenuItem.GetImageSize 3616

RibbonActnCtrls.TRibbonBaseButtonControl 3616

RibbonActnCtrls.TRibbonBaseButtonControl.AllowHotKeys 3619

RibbonActnCtrls.TRibbonBaseButtonControl.CalcBounds 3619

RibbonActnCtrls.TRibbonBaseButtonControl.CanAddToQuickAccessToolbar 3620

RibbonActnCtrls.TRibbonBaseButtonControl.Click 3620

RibbonActnCtrls.TRibbonBaseButtonControl.Create 3617

RibbonActnCtrls.TRibbonBaseButtonControl.Destroy 3618

RibbonActnCtrls.TRibbonBaseButtonControl.DragDrop 3620

RibbonActnCtrls.TRibbonBaseButtonControl.DrawBackground 3621

RibbonActnCtrls.TRibbonBaseButtonControl.DrawDown 3626

RibbonActnCtrls.TRibbonBaseButtonControl.DrawGlyph 3621

RibbonActnCtrls.TRibbonBaseButtonControl.DrawSeparator 3621

RibbonActnCtrls.TRibbonBaseButtonControl.DrawShadowedText 3622

RibbonActnCtrls.TRibbonBaseButtonControl.DrawText 3622

RibbonActnCtrls.TRibbonBaseButtonControl.GetImageSize 3622

RibbonActnCtrls.TRibbonBaseButtonControl.GetKeyTip 3622

RibbonActnCtrls.TRibbonBaseButtonControl.HandleKeyTip 3623

RibbonActnCtrls.TRibbonBaseButtonControl.HideKeyTip 3623

RibbonActnCtrls.TRibbonBaseButtonControl.IsButton 3624

RibbonActnCtrls.TRibbonBaseButtonControl.IsRibbonAvailable 3624

RibbonActnCtrls.TRibbonBaseButtonControl.KeyTipClick 3624

RibbonActnCtrls.TRibbonBaseButtonControl.RibbonStyle 3624

RibbonActnCtrls.TRibbonBaseButtonControl.SetKeyTip 3625

RibbonActnCtrls.TRibbonBaseButtonControl.ShowKeyTip 3625

RibbonActnCtrls.TRibbonCheckBoxControl 3626

RibbonActnCtrls.TRibbonCheckBoxControl.CalcBounds 3627

RibbonActnCtrls.TRibbonCheckBoxControl.DrawBackground 3628

RibbonActnCtrls.TRibbonCheckBoxControl.DrawGlyph 3628

RibbonActnCtrls.TRibbonComboBox 3628

RibbonActnCtrls.TRibbonComboBox.Alignment 3631

RibbonActnCtrls.TRibbonComboBox.AutoSelect 3631

RibbonActnCtrls.TRibbonComboBox.CharCase 3632

RibbonActnCtrls.TRibbonComboBox.Font 3632

RibbonActnCtrls.TRibbonComboBox.HideSelection 3633

RibbonActnCtrls.TRibbonComboBox.ImeMode 3634

RibbonActnCtrls.TRibbonComboBox.ImeName 3634
RibbonActnCtrls.TRibbonComboBox.Items 3635
RibbonActnCtrls.TRibbonComboBox.MaxLength 3635
RibbonActnCtrls.TRibbonComboBox.NumbersOnly 3636
RibbonActnCtrls.TRibbonComboBox.OnChange 3648
RibbonActnCtrls.TRibbonComboBox.OnClick 3649
RibbonActnCtrls.TRibbonComboBox.OnContextPopup 3651
RibbonActnCtrls.TRibbonComboBox.OnDblClick 3651
RibbonActnCtrls.TRibbonComboBox.OnDragDrop 3652
RibbonActnCtrls.TRibbonComboBox.OnDragOver 3655
RibbonActnCtrls.TRibbonComboBox.OnEndDock 3658
RibbonActnCtrls.TRibbonComboBox.OnEndDrag 3658
RibbonActnCtrls.TRibbonComboBox.OnEnter 3659
RibbonActnCtrls.TRibbonComboBox.OnExit 3662
RibbonActnCtrls.TRibbonComboBox.OnKeyDown 3666
RibbonActnCtrls.TRibbonComboBox.OnKeyPress 3669
RibbonActnCtrls.TRibbonComboBox.OnKeyUp 3670
RibbonActnCtrls.TRibbonComboBox.OnMouseActivate 3672
RibbonActnCtrls.TRibbonComboBox.OnMouseDown 3673
RibbonActnCtrls.TRibbonComboBox.OnMouseEnter 3675
RibbonActnCtrls.TRibbonComboBox.OnMouseLeave 3676
RibbonActnCtrls.TRibbonComboBox.OnMouseMove 3676
RibbonActnCtrls.TRibbonComboBox.OnMouseUp 3678
RibbonActnCtrls.TRibbonComboBox.OnStartDock 3680
RibbonActnCtrls.TRibbonComboBox.OnStartDrag 3681
RibbonActnCtrls.TRibbonComboBox.ParentFont 3636
RibbonActnCtrls.TRibbonComboBox.PasswordChar 3636
RibbonActnCtrls.TRibbonComboBox.PopupMenu 3637
RibbonActnCtrls.TRibbonComboBox.ReadOnly 3642
RibbonActnCtrls.TRibbonComboBox.ShowHint 3642
RibbonActnCtrls.TRibbonComboBox.TabOrder 3644
RibbonActnCtrls.TRibbonComboBox.TabStop 3644
RibbonActnCtrls.TRibbonComboBox.Text 3645
RibbonActnCtrls.TRibbonComboBox.TextHint 3646
RibbonActnCtrls.TRibbonComboBox.Visible 3646
RibbonActnCtrls.TRibbonComboBoxItem 3682
RibbonActnCtrls.TRibbonComboBoxItem.CalcBounds 3683
RibbonActnCtrls.TRibbonComboBoxItem.DrawBackground 3683
RibbonActnCtrls.TRibbonComboBoxItem.DrawText 3683
RibbonActnCtrls.TRibbonComboControl 3684
RibbonActnCtrls.TRibbonComboControl.CalcBounds 3687
RibbonActnCtrls.TRibbonComboControl.CanAddToQuickAccess Toolbar
3687
RibbonActnCtrls.TRibbonComboControl.Click 3688
RibbonActnCtrls.TRibbonComboControl.CMBidimodechanged 3688
RibbonActnCtrls.TRibbonComboControl.CMMouseenter 3688
RibbonActnCtrls.TRibbonComboControl.CMMouseleave 3688
RibbonActnCtrls.TRibbonComboControl.CMVisiblechanged 3689
RibbonActnCtrls.TRibbonComboControl.ComboBox 3697
RibbonActnCtrls.TRibbonComboControl.ComboClick 3689
RibbonActnCtrls.TRibbonComboControl.ContainedControl 3697
RibbonActnCtrls.TRibbonComboControl.Create 3685
RibbonActnCtrls.TRibbonComboControl.Destroy 3686
RibbonActnCtrls.TRibbonComboControl.DrawBackground 3689
RibbonActnCtrls.TRibbonComboControl.DrawFrame 3689
RibbonActnCtrls.TRibbonComboControl.DropDownClick 3690
RibbonActnCtrls.TRibbonComboControl.GetCaptionWidth 3690
RibbonActnCtrls.TRibbonComboControl.GetContainedControl 3690
RibbonActnCtrls.TRibbonComboControl.GetGlyphWidth 3690
RibbonActnCtrls.TRibbonComboControl.GetKeyTip 3690
RibbonActnCtrls.TRibbonComboControl.GetLabelWidth 3691
RibbonActnCtrls.TRibbonComboControl.HandleKeyTip 3691
RibbonActnCtrls.TRibbonComboControl.HideKeyTip 3692
RibbonActnCtrls.TRibbonComboControl.KeyTipClick 3692
RibbonActnCtrls.TRibbonComboControl.MouseDown 3692
RibbonActnCtrls.TRibbonComboControl.MouseInControl 3697
RibbonActnCtrls.TRibbonComboControl.Notification 3693
RibbonActnCtrls.TRibbonComboControl.Paint 3693
RibbonActnCtrls.TRibbonComboControl.RibbonStyle 3694
RibbonActnCtrls.TRibbonComboControl.SetActionClient 3694
RibbonActnCtrls.TRibbonComboControl.SetBounds 3694
RibbonActnCtrls.TRibbonComboControl.SetContainedControl 3694
RibbonActnCtrls.TRibbonComboControl.SetDragMode 3695
RibbonActnCtrls.TRibbonComboControl.SetKeyTip 3695
RibbonActnCtrls.TRibbonComboControl.SetParent 3695

RibbonActnCtrls.TRibbonComboControl.ShowComboItems 3696
RibbonActnCtrls.TRibbonComboControl.ShowKeyTip 3696
RibbonActnCtrls.TRibbonCustomControl 3698
RibbonActnCtrls.TRibbonCustomControl.CalcBounds 3700
RibbonActnCtrls.TRibbonCustomControl.CanAddToQuickAccessToolbar 3701
RibbonActnCtrls.TRibbonCustomControl.Click 3701
RibbonActnCtrls.TRibbonCustomControl.CMVisiblechanged 3701
RibbonActnCtrls.TRibbonCustomControl.ContainedControl 3710
RibbonActnCtrls.TRibbonCustomControl.Create 3699
RibbonActnCtrls.TRibbonCustomControl.Destroy 3699
RibbonActnCtrls.TRibbonCustomControl.DrawBackground 3702
RibbonActnCtrls.TRibbonCustomControl.GetCaptionWidth 3702
RibbonActnCtrls.TRibbonCustomControl.GetContainedControl 3702
RibbonActnCtrls.TRibbonCustomControl.GetGlyphWidth 3702
RibbonActnCtrls.TRibbonCustomControl.GetKeyTip 3703
RibbonActnCtrls.TRibbonCustomControl.GetLabelWidth 3703
RibbonActnCtrls.TRibbonCustomControl.HandleKeyTip 3703
RibbonActnCtrls.TRibbonCustomControl.HideKeyTip 3704
RibbonActnCtrls.TRibbonCustomControl.KeyTipClick 3704
RibbonActnCtrls.TRibbonCustomControl.Notification 3704
RibbonActnCtrls.TRibbonCustomControl.Paint 3705
RibbonActnCtrls.TRibbonCustomControl.SetActionClient 3705
RibbonActnCtrls.TRibbonCustomControl.SetBounds 3705
RibbonActnCtrls.TRibbonCustomControl.SetContainedControl 3708
RibbonActnCtrls.TRibbonCustomControl.SetDragMode 3708
RibbonActnCtrls.TRibbonCustomControl.SetKeyTip 3708
RibbonActnCtrls.TRibbonCustomControl.SetParent 3709
RibbonActnCtrls.TRibbonCustomControl.ShowKeyTip 3709
RibbonActnCtrls.TRibbonCustomizeItem 3710
RibbonActnCtrls.TRibbonCustomizeItem.CalcBounds 3711
RibbonActnCtrls.TRibbonCustomizeItem.DrawBackground 3712
RibbonActnCtrls.TRibbonCustomizeItem.DrawGlyph 3712
RibbonActnCtrls.TRibbonCustomizeItem.RibbonStyle 3712
RibbonActnCtrls.TRibbonCustomStatusControl 3713
RibbonActnCtrls.TRibbonCustomStatusControl.CalcBounds 3714
RibbonActnCtrls.TRibbonCustomStatusControl.DrawBackground 3715
RibbonActnCtrls.TRibbonDropDownButton 3715
RibbonActnCtrls.TRibbonDropDownButton.CalcBounds 3717
RibbonActnCtrls.TRibbonDropDownButton.CalcLayout 3717
RibbonActnCtrls.TRibbonDropDownButton.Click 3717
RibbonActnCtrls.TRibbonDropDownButton.DoDropDown 3718
RibbonActnCtrls.TRibbonDropDownButton.DrawArrow 3718
RibbonActnCtrls.TRibbonDropDownButton.DrawBackground 3718
RibbonActnCtrls.TRibbonDropDownButton.DrawShadowedText 3718
RibbonActnCtrls.TRibbonDropDownButton.DrawText 3719
RibbonActnCtrls.TRibbonDropDownButton.GetArrowPosition 3719
RibbonActnCtrls.TRibbonDropDownButton.GetPopupClass 3719
RibbonActnCtrls.TRibbonDropDownButton.KeyTipClick 3719
RibbonActnCtrls.TRibbonGallery 3720
RibbonActnCtrls.TRibbonGalleryControl 3720
RibbonActnCtrls.TRibbonGalleryControl.CanAddToQuickAccessToolbar 3721
RibbonActnCtrls.TRibbonGalleryControl.ContainedControl 3722
RibbonActnCtrls.TRibbonGalleryControl.Gallery 3722
RibbonActnCtrls.TRibbonGalleryControl.GetContainedControl 3721
RibbonActnCtrls.TRibbonGalleryControl.SetContainedControl 3721
RibbonActnCtrls.TRibbonGalleryControlMenuItem 3722
RibbonActnCtrls.TRibbonGalleryControlMenuItem.CalcBounds 3723
RibbonActnCtrls.TRibbonGalleryControlMenuItem.CanAddToQuickAccessToolbar 3724
RibbonActnCtrls.TRibbonGalleryControlMenuItem.ContainedControl 3728
RibbonActnCtrls.TRibbonGalleryControlMenuItem.GetContainedControl 3724
RibbonActnCtrls.TRibbonGalleryControlMenuItem.Paint 3724
RibbonActnCtrls.TRibbonGalleryControlMenuItem.SetBounds 3725
RibbonActnCtrls.TRibbonGalleryControlMenuItem.SetContained

Control
3727

RibbonActnCtrls.TRibbonGalleryControlMenuItem.SetParent
3727

RibbonActnCtrls.TRibbonGalleryItem 3728

RibbonActnCtrls.TRibbonGalleryItem.CalcBounds 3730

RibbonActnCtrls.TRibbonGalleryItem.CalcLayout 3730

RibbonActnCtrls.TRibbonGalleryMenuItem 3730

RibbonActnCtrls.TRibbonGalleryMenuItem.CalcBounds 3731

RibbonActnCtrls.TRibbonGalleryMenuItem.DrawBackground
3732

RibbonActnCtrls.TRibbonGalleryMenuItem.DrawGlyph 3732

RibbonActnCtrls.TRibbonGalleryMenuItem.DrawShadowedText
3732

RibbonActnCtrls.TRibbonGalleryMenuItem.DrawText 3733

RibbonActnCtrls.TRibbonGallerySplitButton 3733

RibbonActnCtrls.TRibbonGallerySplitButton.GetPopupClass
3735

RibbonActnCtrls.TRibbonGripItem 3735

RibbonActnCtrls.TRibbonGripItem.CalcBounds 3737

RibbonActnCtrls.TRibbonGripItem.Create 3736

RibbonActnCtrls.TRibbonGripItem.GalleryResize 3738

RibbonActnCtrls.TRibbonGripItem.Paint 3737

RibbonActnCtrls.TRibbonGripItem.RibbonStyle 3738

RibbonActnCtrls.TRibbonGroupDropDown 3738

RibbonActnCtrls.TRibbonGroupDropDown.CloseUp 3740

RibbonActnCtrls.TRibbonGroupDropDown.DoDropDown 3740

RibbonActnCtrls.TRibbonGroupDropDown.DrawArrow 3741

RibbonActnCtrls.TRibbonGroupDropDown.DrawBackground
3741

RibbonActnCtrls.TRibbonGroupDropDown.GetParentControl
3741

RibbonActnCtrls.TRibbonGroupDropDown.GetTopPosition 3741

RibbonActnCtrls.TRibbonGroupDropDown.Group 3742

RibbonActnCtrls.TRibbonLargeButtonControl 3742

RibbonActnCtrls.TRibbonLargeButtonControl.CalcBounds 3744

RibbonActnCtrls.TRibbonLargeButtonControl.CalcLayout 3744

RibbonActnCtrls.TRibbonLargeButtonControl.DrawBackground
3744

RibbonActnCtrls.TRibbonLargeButtonControl.DrawShadowedTe
xt
3745

RibbonActnCtrls.TRibbonLargeButtonControl.DrawText 3745

RibbonActnCtrls.TRibbonLargeSplitButton 3745

RibbonActnCtrls.TRibbonLargeSplitButton.CalcBounds 3747

RibbonActnCtrls.TRibbonLargeSplitButton.CalcLayout 3748

RibbonActnCtrls.TRibbonLargeSplitButton.Click 3748

RibbonActnCtrls.TRibbonLargeSplitButton.CMMouseleave 3748

RibbonActnCtrls.TRibbonLargeSplitButton.CursorInSplit 3752

RibbonActnCtrls.TRibbonLargeSplitButton.DoSplitButtonClick
3748

RibbonActnCtrls.TRibbonLargeSplitButton.DrawArrow 3749

RibbonActnCtrls.TRibbonLargeSplitButton.DrawBackground
3749

RibbonActnCtrls.TRibbonLargeSplitButton.GetArrowPosition
3749

RibbonActnCtrls.TRibbonLargeSplitButton.InSplit 3749

RibbonActnCtrls.TRibbonLargeSplitButton.KeyTipClick 3750

RibbonActnCtrls.TRibbonLargeSplitButton.MouseDown 3750

RibbonActnCtrls.TRibbonLargeSplitButton.MouseMove 3750

RibbonActnCtrls.TRibbonLargeSplitButton.MouseUp 3751

RibbonActnCtrls.TRibbonLargeSplitButton.UpdateSplitBounds
3751

RibbonActnCtrls.TRibbonMainMenuItem 3752

RibbonActnCtrls.TRibbonMainMenuItem.DrawShadowedText
3753

RibbonActnCtrls.TRibbonMainMenuItem.DrawText 3753

RibbonActnCtrls.TRibbonMainMenuSeparator 3754

RibbonActnCtrls.TRibbonMainMenuSeparator.CalcBounds 3755

RibbonActnCtrls.TRibbonMenuItem 3755

RibbonActnCtrls.TRibbonMenuItem.CalcBounds 3757

RibbonActnCtrls.TRibbonMenuItem.CalcLayout 3757

RibbonActnCtrls.TRibbonMenuItem.Click 3757

RibbonActnCtrls.TRibbonMenuItem.Create 3756

RibbonActnCtrls.TRibbonMenuItem.DrawBackground 3758

RibbonActnCtrls.TRibbonMenuItem.DrawBanner 3760

RibbonActnCtrls.TRibbonMenuItem.DrawGlyph 3758

RibbonActnCtrls.TRibbonMenuItem.DrawSelectedBackground
3758

RibbonActnCtrls.TRibbonMenuItem.DrawSeparator 3759

RibbonActnCtrls.TRibbonMenuItem.DrawShadowedText 3759

RibbonActnCtrls.TRibbonMenuItem.DrawText 3759

RibbonActnCtrls.TRibbonMenuItem.GetShowShortCut 3759

RibbonActnCtrls.TRibbonMenuItem.RibbonStyle 3760

RibbonActnCtrls.TRibbonMenuSeparatorItem 3760
RibbonActnCtrls.TRibbonMenuSeparatorItem.CalcBounds 3761
RibbonActnCtrls.TRibbonMenuSeparatorItem.DrawSeparator 3762
RibbonActnCtrls.TRibbonMenuSmallButtonControl 3762
RibbonActnCtrls.TRibbonMenuSmallButtonControl.Create 3763
RibbonActnCtrls.TRibbonMenuSmallButtonControl.MouseUp 3764
RibbonActnCtrls.TRibbonOptionButtonControl 3765
RibbonActnCtrls.TRibbonOptionButtonControl.DrawBackground 3766
RibbonActnCtrls.TRibbonQATScrollButton 3767
RibbonActnCtrls.TRibbonQATScrollButton.CalcBounds 3768
RibbonActnCtrls.TRibbonQATScrollButton.Click 3768
RibbonActnCtrls.TRibbonQATScrollButton.CMMouseEnter 3769
RibbonActnCtrls.TRibbonQATScrollButton.CMMouseLeave 3769
RibbonActnCtrls.TRibbonQATScrollButton.CursorInBtn 3771
RibbonActnCtrls.TRibbonQATScrollButton.DrawArrows 3769
RibbonActnCtrls.TRibbonQATScrollButton.DrawBackground 3769
RibbonActnCtrls.TRibbonQATScrollButton.DrawDown 3771
RibbonActnCtrls.TRibbonQATScrollButton.MouseMove 3770
RibbonActnCtrls.TRibbonQATScrollButton.RibbonStyle 3770
RibbonActnCtrls.TRibbonQATScrollButton.ScrollMode 3771
RibbonActnCtrls.TRibbonRadioButtonControl 3771
RibbonActnCtrls.TRibbonRadioButtonControl.CalcBounds 3773
RibbonActnCtrls.TRibbonRadioButtonControl.DrawBackground 3773
RibbonActnCtrls.TRibbonRadioButtonControl.DrawGlyph 3773
RibbonActnCtrls.TRibbonRecentButtonControl 3774
RibbonActnCtrls.TRibbonRecentButtonControl.AllowHotKeys 3775
RibbonActnCtrls.TRibbonRichContentItem 3776
RibbonActnCtrls.TRibbonRichContentItem.CalcBounds 3777
RibbonActnCtrls.TRibbonRichContentItem.CanCalculateHeight 3778
RibbonActnCtrls.TRibbonRichContentItem.Create 3777
RibbonActnCtrls.TRibbonRichContentItem.DrawBackground 3778
RibbonActnCtrls.TRibbonRichContentItem.DrawGlyph 3778
RibbonActnCtrls.TRibbonRichContentItem.DrawShadowedText 3779
RibbonActnCtrls.TRibbonRichContentItem.DrawText 3779
RibbonActnCtrls.TRibbonRichContentItem.GetItemHeight 3779
RibbonActnCtrls.TRibbonRichContentItem.GetItemWidth 3780
RibbonActnCtrls.TRibbonRichContentItem.GetMenuProps 3780
RibbonActnCtrls.TRibbonRichContentMainMenuItem 3780
RibbonActnCtrls.TRibbonRichContentMainMenuItem.CalcBounds 3782
RibbonActnCtrls.TRibbonRichContentMainMenuItem.CalcLayout 3782
RibbonActnCtrls.TRibbonRichContentMainMenuItem.Create 3781
RibbonActnCtrls.TRibbonRichContentMainMenuItem.DrawGlyph 3782
RibbonActnCtrls.TRibbonRichContentMainMenuItem.DrawText 3783
RibbonActnCtrls.TRibbonRichContentMainMenuItem.GetImageSize 3783
RibbonActnCtrls.TRibbonRichContentMainMenuItem.GetShowShortCut 3783
RibbonActnCtrls.TRibbonSeparator 3784
RibbonActnCtrls.TRibbonSeparator.CalcBounds 3785
RibbonActnCtrls.TRibbonSeparator.Create 3784
RibbonActnCtrls.TRibbonSeparator.DrawBackground 3785
RibbonActnCtrls.TRibbonSeparator.DrawGlyph 3786
RibbonActnCtrls.TRibbonSeparator.DrawShadowedText 3786
RibbonActnCtrls.TRibbonSeparator.DrawText 3786
RibbonActnCtrls.TRibbonSeparator.RibbonStyle 3787
RibbonActnCtrls.TRibbonSmallButtonControl 3787
RibbonActnCtrls.TRibbonSmallButtonControl.CalcBounds 3788
RibbonActnCtrls.TRibbonSmallSplitButton 3789
RibbonActnCtrls.TRibbonSmallSplitButton.CalcBounds 3792
RibbonActnCtrls.TRibbonSmallSplitButton.CalcLayout 3792
RibbonActnCtrls.TRibbonSmallSplitButton.Click 3793
RibbonActnCtrls.TRibbonSmallSplitButton.CMMouseLeave 3793
RibbonActnCtrls.TRibbonSmallSplitButton.Create 3790
RibbonActnCtrls.TRibbonSmallSplitButton.CursorInSplit 3796
RibbonActnCtrls.TRibbonSmallSplitButton.DrawBackground 3793

RibbonActnCtrls.TRibbonSmallSplitButton.FCursorInSplit 3791
RibbonActnCtrls.TRibbonSmallSplitButton.FSplitRect 3791
RibbonActnCtrls.TRibbonSmallSplitButton.GetArrowPosition 3794
RibbonActnCtrls.TRibbonSmallSplitButton.InSplit 3794
RibbonActnCtrls.TRibbonSmallSplitButton.KeyTipClick 3794
RibbonActnCtrls.TRibbonSmallSplitButton.MouseDown 3794
RibbonActnCtrls.TRibbonSmallSplitButton.MouseMove 3795
RibbonActnCtrls.TRibbonSmallSplitButton.UpdateSplitBounds 3795
RibbonActnCtrls.TRibbonSpinButton 3796
RibbonActnCtrls.TRibbonSpinButton.Align 3801
RibbonActnCtrls.TRibbonSpinButton.Anchors 3804
RibbonActnCtrls.TRibbonSpinButton.Constraints 3805
RibbonActnCtrls.TRibbonSpinButton.Control 3798
RibbonActnCtrls.TRibbonSpinButton.Create 3797
RibbonActnCtrls.TRibbonSpinButton.Ctl3D 3806
RibbonActnCtrls.TRibbonSpinButton.DownGlyph 3806
RibbonActnCtrls.TRibbonSpinButton.DownNumGlyphs 3806
RibbonActnCtrls.TRibbonSpinButton.DragCursor 3807
RibbonActnCtrls.TRibbonSpinButton.DragKind 3807
RibbonActnCtrls.TRibbonSpinButton.DragMode 3807
RibbonActnCtrls.TRibbonSpinButton.Enabled 3809
RibbonActnCtrls.TRibbonSpinButton.FocusControl 3809
RibbonActnCtrls.TRibbonSpinButton.KeyDown 3798
RibbonActnCtrls.TRibbonSpinButton.Loaded 3799
RibbonActnCtrls.TRibbonSpinButton.Notification 3799
RibbonActnCtrls.TRibbonSpinButton.OnDownClick 3820
RibbonActnCtrls.TRibbonSpinButton.OnDragDrop 3820
RibbonActnCtrls.TRibbonSpinButton.OnDragOver 3823
RibbonActnCtrls.TRibbonSpinButton.OnEndDock 3827
RibbonActnCtrls.TRibbonSpinButton.OnEndDrag 3827
RibbonActnCtrls.TRibbonSpinButton.OnEnter 3828
RibbonActnCtrls.TRibbonSpinButton.OnExit 3831
RibbonActnCtrls.TRibbonSpinButton.OnStartDock 3835
RibbonActnCtrls.TRibbonSpinButton.OnStartDrag 3835
RibbonActnCtrls.TRibbonSpinButton.OnUpClick 3836
RibbonActnCtrls.TRibbonSpinButton.ParentCtl3D 3810
RibbonActnCtrls.TRibbonSpinButton.ParentShowHint 3810
RibbonActnCtrls.TRibbonSpinButton.PopupMenu 3810
RibbonActnCtrls.TRibbonSpinButton.SetBounds 3800
RibbonActnCtrls.TRibbonSpinButton.ShowHint 3815
RibbonActnCtrls.TRibbonSpinButton.TabOrder 3817
RibbonActnCtrls.TRibbonSpinButton.TabStop 3818
RibbonActnCtrls.TRibbonSpinButton.UpGlyph 3818
RibbonActnCtrls.TRibbonSpinButton.UpNumGlyphs 3818
RibbonActnCtrls.TRibbonSpinButton.Visible 3818
RibbonActnCtrls.TRibbonSpinEdit 3836
RibbonActnCtrls.TRibbonSpinEdit.Anchors 3845
RibbonActnCtrls.TRibbonSpinEdit.AutoSelect 3846
RibbonActnCtrls.TRibbonSpinEdit.AutoSize 3846
RibbonActnCtrls.TRibbonSpinEdit.Button 3847
RibbonActnCtrls.TRibbonSpinEdit.CMMouseenter 3839
RibbonActnCtrls.TRibbonSpinEdit.CMMouseleave 3840
RibbonActnCtrls.TRibbonSpinEdit.Color 3847
RibbonActnCtrls.TRibbonSpinEdit.Constraints 3848
RibbonActnCtrls.TRibbonSpinEdit.Create 3838
RibbonActnCtrls.TRibbonSpinEdit.CreateParams 3840
RibbonActnCtrls.TRibbonSpinEdit.CreateWnd 3840
RibbonActnCtrls.TRibbonSpinEdit.Ctl3D 3849
RibbonActnCtrls.TRibbonSpinEdit.Destroy 3838
RibbonActnCtrls.TRibbonSpinEdit.DownClick 3841
RibbonActnCtrls.TRibbonSpinEdit.DragCursor 3849
RibbonActnCtrls.TRibbonSpinEdit.DragMode 3850
RibbonActnCtrls.TRibbonSpinEdit.EditorEnabled 3851
RibbonActnCtrls.TRibbonSpinEdit.Enabled 3851
RibbonActnCtrls.TRibbonSpinEdit.Font 3852
RibbonActnCtrls.TRibbonSpinEdit.GetChildren 3841
RibbonActnCtrls.TRibbonSpinEdit.Increment 3853
RibbonActnCtrls.TRibbonSpinEdit.IsValidChar 3841
RibbonActnCtrls.TRibbonSpinEdit.KeyDown 3842
RibbonActnCtrls.TRibbonSpinEdit.KeyPress 3842
RibbonActnCtrls.TRibbonSpinEdit.MaxLength 3853
RibbonActnCtrls.TRibbonSpinEdit.MaxValue 3854
RibbonActnCtrls.TRibbonSpinEdit.MinValue 3854
RibbonActnCtrls.TRibbonSpinEdit.OnChange 3866
RibbonActnCtrls.TRibbonSpinEdit.OnClick 3867
RibbonActnCtrls.TRibbonSpinEdit.OnDblClick 3868
RibbonActnCtrls.TRibbonSpinEdit.OnDragDrop 3869

RibbonActnCtrls.TRibbonSpinEdit.OnDragOver 3872
RibbonActnCtrls.TRibbonSpinEdit.OnEndDrag 3875
RibbonActnCtrls.TRibbonSpinEdit.OnEnter 3876
RibbonActnCtrls.TRibbonSpinEdit.OnExit 3878
RibbonActnCtrls.TRibbonSpinEdit.OnKeyDown 3882
RibbonActnCtrls.TRibbonSpinEdit.OnKeyPress 3886
RibbonActnCtrls.TRibbonSpinEdit.OnKeyUp 3887
RibbonActnCtrls.TRibbonSpinEdit.OnMouseDown 3889
RibbonActnCtrls.TRibbonSpinEdit.OnMouseMove 3891
RibbonActnCtrls.TRibbonSpinEdit.OnMouseUp 3893
RibbonActnCtrls.TRibbonSpinEdit.OnStartDrag 3896
RibbonActnCtrls.TRibbonSpinEdit.ParentColor 3854
RibbonActnCtrls.TRibbonSpinEdit.ParentCtl3D 3855
RibbonActnCtrls.TRibbonSpinEdit.ParentFont 3855
RibbonActnCtrls.TRibbonSpinEdit.ParentShowHint 3855
RibbonActnCtrls.TRibbonSpinEdit.PopupMenu 3856
RibbonActnCtrls.TRibbonSpinEdit.ReadOnly 3861
RibbonActnCtrls.TRibbonSpinEdit.SetEnabled 3843
RibbonActnCtrls.TRibbonSpinEdit.SetParent 3843
RibbonActnCtrls.TRibbonSpinEdit.ShowHint 3861
RibbonActnCtrls.TRibbonSpinEdit.TabOrder 3863
RibbonActnCtrls.TRibbonSpinEdit.TabStop 3864
RibbonActnCtrls.TRibbonSpinEdit.UpClick 3844
RibbonActnCtrls.TRibbonSpinEdit.Value 3864
RibbonActnCtrls.TRibbonSpinEdit.Visible 3864
RibbonActnCtrls.TRibbonSpinEdit.WMKillFocus 3844
RibbonActnCtrls.TRibbonSplitMenuItem 3896
RibbonActnCtrls.TRibbonSplitMenuItem.CalcBounds 3899
RibbonActnCtrls.TRibbonSplitMenuItem.Click 3899
RibbonActnCtrls.TRibbonSplitMenuItem.Create 3898
RibbonActnCtrls.TRibbonSplitMenuItem.DrawBackground 3900
RibbonActnCtrls.TRibbonSplitMenuItem.DrawSelectedBackground 3900
RibbonActnCtrls.TRibbonSplitMenuItem.FMouseInSplit 3898
RibbonActnCtrls.TRibbonSplitMenuItem.FSplitRect 3899
RibbonActnCtrls.TRibbonSplitMenuItem.IsInSplit 3900
RibbonActnCtrls.TRibbonSplitMenuItem.MouseInSplit 3902
RibbonActnCtrls.TRibbonSplitMenuItem.MouseMove 3901
RibbonActnCtrls.TRibbonSplitMenuItem.UpdateSplitBounds 3901
RibbonActnCtrls.TRibbonStyleMenuButton 3902
RibbonActnCtrls.TRibbonStyleMenuButton.DrawBackground 3903
RibbonActnCtrls.TRibbonStyleMenuButton.RibbonStyle 3903
RibbonActnCtrls.TRibbonTextControl 3903
RibbonActnCtrls.TRibbonTextControl.CalcBounds 3904
RibbonActnCtrls.TRibbonTextControl.DrawBackground 3904
RibbonActnCtrls.TRibbonTextControl.DrawText 3905
RibbonActnCtrls.TRibbonToolScrollButton 3905
RibbonActnCtrls.TRibbonToolScrollButton.DrawArrows 3906
RibbonActnCtrls.TRibbonToolScrollButton.DrawBackground 3906
RibbonActnCtrls.TSpinButtonStyle 3930
RibbonActnCtrls.TSpinButtonStyle enumeration 3930
RibbonActnCtrls.TTimeBtnState 3931
RibbonActnCtrls.TTimeBtnState type 3931
RibbonActnCtrls.TTimerSpeedButton 3906
RibbonActnCtrls.TTimerSpeedButton.ButtonStyle 3910
RibbonActnCtrls.TTimerSpeedButton.CMMouseenter 3908
RibbonActnCtrls.TTimerSpeedButton.CMMouseleave 3908
RibbonActnCtrls.TTimerSpeedButton.Create 3907
RibbonActnCtrls.TTimerSpeedButton.Destroy 3908
RibbonActnCtrls.TTimerSpeedButton.MouseDown 3909
RibbonActnCtrls.TTimerSpeedButton.MouseInControl 3910
RibbonActnCtrls.TTimerSpeedButton.MouseUp 3909
RibbonActnCtrls.TTimerSpeedButton.Paint 3910
RibbonActnCtrls.TTimerSpeedButton.TimeBtnState 3911
RibbonActnCtrls.UM_FADECONTROL 3932
RibbonActnCtrls.UM_FADECONTROL constant 3932
RibbonActnMenus 3933
RibbonActnMenus namespace 3933
Classes 3934
RibbonActnMenus.TOptionItem 3934
RibbonActnMenus.TOptionItem.Action 3936
RibbonActnMenus.TOptionItem.Assign 3935
RibbonActnMenus.TOptionItem.Caption 3937
RibbonActnMenus.TOptionItem.GetDisplayName 3936
RibbonActnMenus.TOptionItem.Hint 3937
RibbonActnMenus.TOptionItem.KeyTip 3937

RibbonActnMenus.TOptionItem.Tag 3937
RibbonActnMenus.TOptionItems 3938
RibbonActnMenus.TOptionItems.Add 3938
RibbonActnMenus.TOptionItems.ApplicationMenu 3939
RibbonActnMenus.TOptionItems.Insert 3939
RibbonActnMenus.TOptionItems.Items 3940
RibbonActnMenus.TRibbonActionPopupMenu 3940
RibbonActnMenus.TRibbonActionPopupMenu.CreateControl 3941
RibbonActnMenus.TRibbonActionPopupMenu.NCPaint 3941
RibbonActnMenus.TRibbonActionPopupMenu.SelectFirstItem 3942
RibbonActnMenus.TRibbonActionPopupMenu.TrackMenu 3942
RibbonActnMenus.TRibbonApplicationMenuBar 3942
RibbonActnMenus.TRibbonApplicationMenuBar.ActionManager 3950
RibbonActnMenus.TRibbonApplicationMenuBar.Align 3950
RibbonActnMenus.TRibbonApplicationMenuBar.CloseMenu 3945
RibbonActnMenus.TRibbonApplicationMenuBar.Create 3944
RibbonActnMenus.TRibbonApplicationMenuBar.CreateControl 3945
RibbonActnMenus.TRibbonApplicationMenuBar.CreateControls 3946
RibbonActnMenus.TRibbonApplicationMenuBar.Destroy 3944
RibbonActnMenus.TRibbonApplicationMenuBar.DragOver 3946
RibbonActnMenus.TRibbonApplicationMenuBar.DrawBackground 3946
RibbonActnMenus.TRibbonApplicationMenuBar.GetControlClass 3946
RibbonActnMenus.TRibbonApplicationMenuBar.GetRibbon 3947
RibbonActnMenus.TRibbonApplicationMenuBar.Height 3954
RibbonActnMenus.TRibbonApplicationMenuBar.HorzMargin 3955
RibbonActnMenus.TRibbonApplicationMenuBar.Left 3955
RibbonActnMenus.TRibbonApplicationMenuBar.NewPopup 3947
RibbonActnMenus.TRibbonApplicationMenuBar.Notification 3947
RibbonActnMenus.TRibbonApplicationMenuBar.OptionItems 3956
RibbonActnMenus.TRibbonApplicationMenuBar.Paint 3948
RibbonActnMenus.TRibbonApplicationMenuBar.RecentItems 3956
RibbonActnMenus.TRibbonApplicationMenuBar.RibbonStyle 3948
RibbonActnMenus.TRibbonApplicationMenuBar.SelectApplicationButton 3948
RibbonActnMenus.TRibbonApplicationMenuBar.SetBounds 3948
RibbonActnMenus.TRibbonApplicationMenuBar.SetRibbon 3949
RibbonActnMenus.TRibbonApplicationMenuBar.ShowKeyTips 3949
RibbonActnMenus.TRibbonApplicationMenuBar.Spacing 3956
RibbonActnMenus.TRibbonApplicationMenuBar.Top 3957
RibbonActnMenus.TRibbonApplicationMenuBar.VertMargin 3958
RibbonActnMenus.TRibbonApplicationMenuBar.VisibleMenu 3959
RibbonActnMenus.TRibbonApplicationMenuBar.VisibleMenuCount 3959
RibbonActnMenus.TRibbonApplicationMenuBar.Width 3959
RibbonActnMenus.TRibbonApplicationMenuBar.WMSysCommand 3949
RibbonActnMenus.TRibbonApplicationPopupMenu 3961
RibbonActnMenus.TRibbonApplicationPopupMenu.AlignControls 3963
RibbonActnMenus.TRibbonApplicationPopupMenu.CMTextchanged 3963
RibbonActnMenus.TRibbonApplicationPopupMenu.Create 3962
RibbonActnMenus.TRibbonApplicationPopupMenu.CreateControls 3963
RibbonActnMenus.TRibbonApplicationPopupMenu.CreateOptionsItems 3964
RibbonActnMenus.TRibbonApplicationPopupMenu.CreateRecentDocuments 3964
RibbonActnMenus.TRibbonApplicationPopupMenu.CreateWnd 3964
RibbonActnMenus.TRibbonApplicationPopupMenu.CustomAlignPosition 3965
RibbonActnMenus.TRibbonApplicationPopupMenu.Destroy

3962
RibbonActnMenus.TRibbonApplicationPopupMenu.DoDropActions 3965
RibbonActnMenus.TRibbonApplicationPopupMenu.DoDropCategory 3966
RibbonActnMenus.TRibbonApplicationPopupMenu.DoDropItem 3966
RibbonActnMenus.TRibbonApplicationPopupMenu.DrawBackground 3966
RibbonActnMenus.TRibbonApplicationPopupMenu.GetBorderHeight 3966
RibbonActnMenus.TRibbonApplicationPopupMenu.GetRibbon 3966
RibbonActnMenus.TRibbonApplicationPopupMenu.NCPaint 3967
RibbonActnMenus.TRibbonApplicationPopupMenu.PositionPopup 3967
RibbonActnMenus.TRibbonApplicationPopupMenu.RibbonStyle 3967
RibbonActnMenus.TRibbonApplicationPopupMenu.SetColorMap 3967
RibbonActnMenus.TRibbonApplicationPopupMenu.SetRibbon 3968
RibbonActnMenus.TRibbonQATCustomizeMenu 3968
RibbonActnMenus.TRibbonQATCustomizeMenu.DisplayShadow 3969
RibbonActnMenus.TRibbonQATCustomizeMenu.DoAddCustomizeItems 3969
RibbonActnMenus.TRibbonQATCustomizeMenu.GetControlClass 3969
RibbonActnMenus.TRibbonQATCustomizeMenu.HideShadow 3969
RibbonActnMenus.TRibbonQATCustomizeMenu.NCPaint 3970
RibbonActnMenus.TRibbonQATCustomizeMenu.PositionPopup 3970
RibbonActnMenus.TRibbonQATCustomizeMenu.VisibleChanging 3970
RibbonActnMenus.TRibbonRecentDocuments 3970
RibbonActnMenus.TRibbonRecentDocuments.Caption 3972
RibbonActnMenus.TRibbonRecentDocuments.Create 3971
RibbonActnMenus.TRibbonRecentDocuments.Paint 3972
RibbonActnMenus.TRibbonStyleMainPopupMenu 3975
RibbonActnMenus.TRibbonStyleMainPopupMenu.Create 3976
RibbonActnMenus.TRibbonStyleMainPopupMenu.NCPaint 3976
RibbonActnMenus.TRibbonStyleMainPopupMenu.PositionPopup 3976
RibbonActnMenus.TRibbonStylePopupMenu 3977
RibbonActnMenus.TRibbonStylePopupMenu.AlignControls 3978
RibbonActnMenus.TRibbonStylePopupMenu.CanAutoSize 3979
RibbonActnMenus.TRibbonStylePopupMenu.Create 3977
RibbonActnMenus.TRibbonStylePopupMenu.CreateControl 3979
RibbonActnMenus.TRibbonStylePopupMenu.Destroy 3978
RibbonActnMenus.TRibbonStylePopupMenu.GalleryResize 3980
RibbonActnMenus.TRibbonStylePopupMenu.WMExitsizemove 3980
RibbonActnMenus.TRibbonStylePopupMenu.WMNCHitTest 3980
RibbonGalleryBar 3981
RibbonGalleryBar namespace 3981
Classes 3982
RibbonGalleryBar.TCustomRibbonGalleryBar 3982
RibbonGalleryBar.TCustomRibbonGalleryBar.Align 3984
RibbonGalleryBar.TCustomRibbonGalleryBar.AlignControls 3983
RibbonGalleryBar.TCustomRibbonGalleryBar.Create 3983
RibbonGalleryBar.TCustomRibbonGalleryBar.CreateControl 3984
RibbonGalleryBar.TCustomRibbonGalleryBar.DoDropCategory 3984
RibbonGalleryBar.TCustomRibbonGalleryBar.ItemsPerRow 3988
RibbonGalleryBar.TCustomRibbonGalleryBar.ShowItemCaptions 3988
RibbonGalleryBar.TCustomRibbonGalleryPopupBar 3988
RibbonGalleryBar.TCustomRibbonGalleryPopupBar.AlignControls 3989
RibbonGalleryBar.TCustomRibbonGalleryPopupBar.CreateControls 3990

RibbonGalleryBar.TCustomRibbonGalleryPopupBar.DoGetControlClass 3990
RibbonGalleryBar.TCustomRibbonGalleryPopupBar.SetColorMap 3990
RibbonGalleryBar.TRibbonGalleryBar 3990
RibbonGalleryBar.TRibbonGalleryBar.Items 3991
RibbonGalleryBar.TRibbonGalleryBar.ItemsPerRow 3992
RibbonGalleryBar.TRibbonGalleryBar.ShowItemCaptions 3992
RibbonGalleryBar.TRibbonGalleryPopupBar 3992
RibbonGalleryBar.TRibbonPopupGalleryBar 3992
RibbonGalleryBar.TRibbonPopupGalleryBar.Assign 3994
RibbonGalleryBar.TRibbonPopupGalleryBar.CreateParams 3994
RibbonGalleryBar.TRibbonPopupGalleryBar.DisplayGallery 3995
RibbonLunaStyleActnCtrls 3997
RibbonLunaStyleActnCtrls namespace 3997
Classes 3998
Variables 4002
RibbonLunaStyleActnCtrls.RibbonLunaStyle 4002
RibbonLunaStyleActnCtrls.RibbonLunaStyle variable 4002
RibbonLunaStyleActnCtrls.TRibbonLunaSkin 3998
RibbonLunaStyleActnCtrls.TRibbonLunaSkin.GetLoadType 3999
RibbonLunaStyleActnCtrls.TRibbonLunaSkin.GetLoadTypeValue 3999
RibbonLunaStyleActnCtrls.TRibbonLunaStyleActionBars 3999
RibbonLunaStyleActnCtrls.TRibbonLunaStyleActionBars.GetColorMapClass 4000
RibbonLunaStyleActnCtrls.TRibbonLunaStyleActionBars.GetSkinClass 4001
RibbonLunaStyleActnCtrls.TRibbonLunaStyleActionBars.GetStyleName 4001
RibbonObsidianStyleActnCtrls 4003
RibbonObsidianStyleActnCtrls namespace 4003
Classes 4004
Variables 4008
RibbonObsidianStyleActnCtrls.RibbonObsidianStyle 4008
RibbonObsidianStyleActnCtrls.RibbonObsidianStyle variable 4008
RibbonObsidianStyleActnCtrls.TRibbonObsidianSkin 4004
RibbonObsidianStyleActnCtrls.TRibbonObsidianSkin.GetLoadType 4005
RibbonObsidianStyleActnCtrls.TRibbonObsidianSkin.GetLoadTypeValue 4005
RibbonObsidianStyleActnCtrls.TRibbonObsidianStyleActionBars 4005
RibbonObsidianStyleActnCtrls.TRibbonObsidianStyleActionBars.GetColorMapClass 4006
RibbonObsidianStyleActnCtrls.TRibbonObsidianStyleActionBars.GetSkinClass 4007
RibbonObsidianStyleActnCtrls.TRibbonObsidianStyleActionBars.GetStyleName 4007
RibbonSilverStyleActnCtrls 4009
RibbonSilverStyleActnCtrls namespace 4009
Classes 4010
Variables 4014
RibbonSilverStyleActnCtrls.RibbonSilverStyle 4014
RibbonSilverStyleActnCtrls.RibbonSilverStyle variable 4014
RibbonSilverStyleActnCtrls.TRibbonSilverSkin 4010
RibbonSilverStyleActnCtrls.TRibbonSilverSkin.GetLoadType 4011
RibbonSilverStyleActnCtrls.TRibbonSilverSkin.GetLoadTypeValue 4011
RibbonSilverStyleActnCtrls.TRibbonSilverStyleActionBars 4011
RibbonSilverStyleActnCtrls.TRibbonSilverStyleActionBars.GetColorMapClass 4012
RibbonSilverStyleActnCtrls.TRibbonSilverStyleActionBars.GetSkinClass 4013
RibbonSilverStyleActnCtrls.TRibbonSilverStyleActionBars.GetStyleName 4013
RibbonSkin
TRibbonStyleActionBars class 4028
RibbonStyle
TCustomRibbonPage class 3434
TOffice2007Button class 3467

TRibbonApplicationMenuBar class 3948
TRibbonApplicationMenuButton class 3609
TRibbonApplicationPopupMenu class 3967
TRibbonBaseButtonControl class 3624
TRibbonComboControl class 3694
TRibbonCustomizeItem class 3712
TRibbonGripItem class 3738
TRibbonGroupScrollButton class 3524
TRibbonMenuItem class 3760
TRibbonQATScrollButton class 3770
TRibbonSeparator class 3787
TRibbonStyleMenuButton class 3903
RibbonStyleActnCtrls 4015
RibbonStyleActnCtrls namespace 4015
Classes 4017
Structs, Records, Enums 4029
Types 4053
RibbonStyleActnCtrls.TAlphaBitmap 4017
RibbonStyleActnCtrls.TAlphaBitmap.Create 4018
RibbonStyleActnCtrls.TCustomRibbonSkin 4018
RibbonStyleActnCtrls.TCustomRibbonSkin.Bitmap 4022
RibbonStyleActnCtrls.TCustomRibbonSkin.Create 4019
RibbonStyleActnCtrls.TCustomRibbonSkin.Destroy 4019
RibbonStyleActnCtrls.TCustomRibbonSkin.DoLoadByFile 4021
RibbonStyleActnCtrls.TCustomRibbonSkin.DoLoadByResource 4021
RibbonStyleActnCtrls.TCustomRibbonSkin.GetLoadType 4021
RibbonStyleActnCtrls.TCustomRibbonSkin.GetLoadTypeValue 4021
RibbonStyleActnCtrls.TCustomRibbonSkin.GetResourceHandle 4021
RibbonStyleActnCtrls.TCustomRibbonSkin.LoadSkin 4022
RibbonStyleActnCtrls.TCustomRibbonSkin.TRibbonSkinLoadType 4020
RibbonStyleActnCtrls.TCustomRibbonSkin.TRibbonSkinLoadType enumeration 4020
RibbonStyleActnCtrls.TCustomRibbonSkinClass 4053
RibbonStyleActnCtrls.TCustomRibbonSkinClass type 4053
RibbonStyleActnCtrls.TRibbonElement 4029
RibbonStyleActnCtrls.TRibbonElement enumeration 4029
RibbonStyleActnCtrls.TRibbonStyleActionBars 4022
RibbonStyleActnCtrls.TRibbonStyleActionBars.Create 4023
RibbonStyleActnCtrls.TRibbonStyleActionBars.Destroy 4023
RibbonStyleActnCtrls.TRibbonStyleActionBars.DrawElement 4025
RibbonStyleActnCtrls.TRibbonStyleActionBars.GetAddRemoveItemClass 4026
RibbonStyleActnCtrls.TRibbonStyleActionBars.GetColorMapClass 4026
RibbonStyleActnCtrls.TRibbonStyleActionBars.GetControlClass 4027
RibbonStyleActnCtrls.TRibbonStyleActionBars.GetPopupClass 4027
RibbonStyleActnCtrls.TRibbonStyleActionBars.GetScrollBtnClass 4027
RibbonStyleActnCtrls.TRibbonStyleActionBars.GetSkinClass 4028
RibbonStyleActnCtrls.TRibbonStyleActionBars.RibbonSkin 4028
RibbonStyleActnCtrls.TSkinApplicationMenu 4030
RibbonStyleActnCtrls.TSkinApplicationMenu enumeration 4030
RibbonStyleActnCtrls.TSkinButton 4032
RibbonStyleActnCtrls.TSkinButton enumeration 4032
RibbonStyleActnCtrls.TSkinCheckBox 4036
RibbonStyleActnCtrls.TSkinCheckBox enumeration 4036
RibbonStyleActnCtrls.TSkinComboBox 4037
RibbonStyleActnCtrls.TSkinComboBox enumeration 4037
RibbonStyleActnCtrls.TSkinForm 4038
RibbonStyleActnCtrls.TSkinForm enumeration 4038
RibbonStyleActnCtrls.TSkinGallery 4040
RibbonStyleActnCtrls.TSkinGallery enumeration 4040
RibbonStyleActnCtrls.TSkinMenu 4041
RibbonStyleActnCtrls.TSkinMenu enumeration 4041
RibbonStyleActnCtrls.TSkinQuickAccessToolbar 4043
RibbonStyleActnCtrls.TSkinQuickAccessToolbar enumeration 4043
RibbonStyleActnCtrls.TSkinRadioButton 4045
RibbonStyleActnCtrls.TSkinRadioButton enumeration 4045
RibbonStyleActnCtrls.TSkinRibbon 4045
RibbonStyleActnCtrls.TSkinRibbon enumeration 4045
RibbonStyleActnCtrls.TSkinRibbonGroup 4046

RibbonStyleActnCtrls.TSkinRibbonGroup enumeration 4046
RibbonStyleActnCtrls.TSkinScrollBar 4048
RibbonStyleActnCtrls.TSkinScrollBar enumeration 4048
RibbonStyleActnCtrls.TSkinSpinEdit 4049
RibbonStyleActnCtrls.TSkinSpinEdit enumeration 4049
RibbonStyleActnCtrls.TSkinStatusBar 4050
RibbonStyleActnCtrls.TSkinStatusBar enumeration 4050
RibbonStyleActnCtrls.TSkinTab 4051
RibbonStyleActnCtrls.TSkinTab enumeration 4051
RightTabScrollButtonRect
TCustomTabData class 3458
Rio 4055
RIO
IRIOAccess interface 4069
Rio namespace 4055
Classes 4056
Constants 4072
Interfaces 4068
Types 4070
Rio.IRIOAccess 4068
Rio.IRIOAccess.GetRIO 4069
Rio.IRIOAccess.RIO 4069
Rio.StubSize 4072
Rio.StubSize constant 4072
Rio.TAfterExecuteEvent 4070
Rio.TAfterExecuteEvent type 4070
Rio.TBeforeExecuteEvent 4070
Rio.TBeforeExecuteEvent type 4070
Rio.TRIO 4056
Rio.TRIO.AfterConstruction 4060
Rio.TRIO.BeforeDestruction 4060
Rio.TRIO.Converter 4063
Rio.TRIO.Create 4058
Rio.TRIO.Destroy 4058
Rio.TRIO.DoAfterExecute 4061
Rio.TRIO.DoBeforeExecute 4061
Rio.TRIO.FConverter 4059
Rio.TRIO.FIID 4059
Rio.TRIO.FWebNode 4059
Rio.TRIO.GetResponseStream 4062
Rio.TRIO.IntfMD 4059
Rio.TRIO.NewInstance 4062
Rio.TRIO.OnAfterExecute 4065
Rio.TRIO.OnBeforeExecute 4066
Rio.TRIO.OnGetAttachment 4067
Rio.TRIO.OnSendAttachment 4067
Rio.TRIO.QueryInterface 4063
Rio.TRIO.RefCount 4064
Rio.TRIO.SOAPHeaders 4064
Rio.TRIO.WebNode 4065
rmCaption enumeration member 3573
rmDown enumeration member 1265
rmFontSize enumeration member 3573
rmFullHeight enumeration member 3573
rmGroupCaptionHeight enumeration member 3573
rmGroupHeight enumeration member 3573
rmHeight enumeration member 3573
rmMinHeight enumeration member 3573
rmNearest enumeration member 1265
rmPageHeight enumeration member 3573
rmQATWidth enumeration member 3573
rmSystemButtonHeight enumeration member 3573
rmSystemButtonWidth enumeration member 3573
rmTabHeight enumeration member 3573
rmTabTop enumeration member 3573
rmTruncate enumeration member 1265
rmUp enumeration member 1265
Rollback
TIBTransaction class 213
RollbackAction enumeration member 482
RollbackAdvise enumeration member 483
RollbackGlobal enumeration member 483
RollbackRetaining
TIBTransaction class 214
RollbackState enumeration member 484
RootKey
TRegistry class 3297
Row

TCustomOutline class 2870
RowAttributes
TCustomDataGrid class 1818
TDataGrid class 1854
RowDelimiter
TIBInputDelimitedFile class 504
TIBOutputDelimitedFile class 511
RowRequest
TCustomProvider class 3113
TCustomResolver class 3134
Rows
TCustomRibbonGroup class 3424
TRibbonGroup class 3505
RowsAffected
TIBCustomDataSet class 105
TIBQuery class 391
TIBSQL class 534
rrAbort enumeration member 3221
rrApply enumeration member 3221
rrIgnore enumeration member 3221
rrMerge enumeration member 3221
rrSkip enumeration member 3221
rtBitmap enumeration member 688
rtCursor enumeration member 688
rtIcon enumeration member 688
Run
TOleContainer class 2548

S

saCommit enumeration member 225
saCommitRetaining enumeration member 225
SafeCallException
TDispatchAppServer class 1290
TLocalAppServer class 3188
samBorder enumeration member 4030
samButton enumeration member 4030
samButtonHover enumeration member 4030
samButtonPressed enumeration member 4030
samButtonShadow enumeration member 4030
samMenuDisabled enumeration member 4030
samMenuHover enumeration member 4030
samMenuNormal enumeration member 4030
samMenuPressed enumeration member 4030
samMenuSplitHover enumeration member 4030
samMenuSplitNormal enumeration member 4030
samMenuSplitPressed enumeration member 4030
samMenuSubMenu enumeration member 4030
saNone enumeration member 225
saRollback enumeration member 225
saRollbackRetaining enumeration member 225
Save
TMediaPlayer class 2328
SaveAsDocument
TOleContainer class 2548
SaveKey
TRegistry class 3279
SaveObject
TOleControl class 2676
SaveToClipboardFormat
TJPEGImage class 1005
SaveToFile
TCustomOutline class 2859
TIBBlobStream class 47
TIBXSQLVAR class 547
TOleContainer class 2550
TSOAPAttachment class 901
SaveToStream
TCustomOutline class 2859
TIBBlobStream class 47
TIBXSQLVAR class 547
TJPEGImage class 1005
TOleContainer class 2550
TSimpleObjectBroker class 2479
TSOAPAttachment class 901
sbArrow enumeration member 4032
sbArrowDisabled enumeration member 4032
sbArrowPressed enumeration member 4032
sbDisabled enumeration member 4045

sbDisabledSelected enumeration member 4045
sbDisabledSeparator enumeration member 4032
sbGroup enumeration member 4032
sbGroupEnd enumeration member 4032
sbGroupEndHover enumeration member 4032
sbGroupEndHoverSelected enumeration member 4032
sbGroupEndInactive enumeration member 4032
sbGroupEndPressed enumeration member 4032
sbGroupEndSelected enumeration member 4032
sbGroupEndSplitHover enumeration member 4032
sbGroupEndSplitHoverSplit enumeration member 4032
sbGroupEndSplitInactive enumeration member 4032
sbGroupEndSplitInactiveSplit enumeration member 4032
sbGroupEndSplitPressed enumeration member 4032
sbGroupEndSplitPressedSplit enumeration member 4032
sbGroupMiddle enumeration member 4032
sbGroupMiddleHover enumeration member 4032
sbGroupMiddleHoverSelected enumeration member 4032
sbGroupMiddleInactive enumeration member 4032
sbGroupMiddlePressed enumeration member 4032
sbGroupMiddleSelected enumeration member 4032
sbGroupMiddleSplitHover enumeration member 4032
sbGroupMiddleSplitHoverSplit enumeration member 4032
sbGroupMiddleSplitInactive enumeration member 4032
sbGroupMiddleSplitInactiveSplit enumeration member 4032
sbGroupMiddleSplitPressed enumeration member 4032
sbGroupMiddleSplitPressedSplit enumeration member 4032
sbGroupSingle enumeration member 4032
sbGroupSingleHover enumeration member 4032
sbGroupSingleHoverSelected enumeration member 4032
sbGroupSingleInactive enumeration member 4032
sbGroupSinglePressed enumeration member 4032
sbGroupSingleSelected enumeration member 4032
sbGroupSingleSplitHover enumeration member 4032
sbGroupSingleSplitHoverSplit enumeration member 4032
sbGroupSingleSplitInactive enumeration member 4032
sbGroupSingleSplitInactiveSplit enumeration member 4032
sbGroupSingleSplitPressed enumeration member 4032
sbGroupSingleSplitPressedSplit enumeration member 4032
sbGroupSplit enumeration member 4032
sbGroupStart enumeration member 4032
sbGroupStartHover enumeration member 4032
sbGroupStartHoverSelected enumeration member 4032
sbGroupStartInactive enumeration member 4032
sbGroupStartInactiveSplit enumeration member 4032
sbGroupStartPressed enumeration member 4032
sbGroupStartSelected enumeration member 4032
sbGroupStartSplitHover enumeration member 4032
sbGroupStartSplitHoverSplit enumeration member 4032
sbGroupStartSplitInactive enumeration member 4032
sbGroupStartSplitInactiveSplit enumeration member 4032
sbGroupStartSplitPressed enumeration member 4032
sbGroupStartSplitPressedSplit enumeration member 4032
sbHover enumeration member 4045
sbHoverSelected enumeration member 4045
sbLarge enumeration member 4032
sbLargeArrowDown enumeration member 4032
sbLargeArrowDownDisabled enumeration member 4032
sbLargeDisabled enumeration member 4032
sbLargeHover enumeration member 4032
sbLargeHoverSelected enumeration member 4032
sbLargePressed enumeration member 4032
sbLargeSelected enumeration member 4032
sbLargeSplit enumeration member 4032
sbLargeSplitActive enumeration member 4032
sbLargeSplitDisabled enumeration member 4032
sbLargeSplitHover enumeration member 4032
sbLargeSplitHoverSelected enumeration member 4032
sbLargeSplitHoverSplit enumeration member 4032
sbLargeSplitPressed enumeration member 4032
sbLargeSplitPressedSplit enumeration member 4032
sbLargeSplitSelected enumeration member 4032
sbNormal enumeration member 4045
sbPressed enumeration member 4045
sbsDown enumeration member 3930
sbSelected enumeration member 4045
sbSeparator enumeration member 4032
sbSmall enumeration member 4032

sbSmallDisabled enumeration member 4032
sbSmallHover enumeration member 4032
sbSmallHoverSelected enumeration member 4032
sbSmallInactive enumeration member 4032
sbSmallPressed enumeration member 4032
sbSmallSelected enumeration member 4032
sbSmallSplit enumeration member 4032
sbSmallSplitDisabled enumeration member 4032
sbSmallSplitHover enumeration member 4032
sbSmallSplitHoverSelected enumeration member 4032
sbSmallSplitHoverSplit enumeration member 4032
sbSmallSplitInactive enumeration member 4032
sbSmallSplitInactiveSplit enumeration member 4032
sbSmallSplitPressed enumeration member 4032
sbSmallSplitPressedSplit enumeration member 4032
sbSmallSplitSelected enumeration member 4032
sbsUp enumeration member 3930
Scale
TJPEGImage class 1010
ScaleFactor
TCustomRibbon class 3398
scButton enumeration member 4037
scButtonHover enumeration member 4037
scButtonIcon enumeration member 4037
scButtonInactive enumeration member 4037
scButtonPressed enumeration member 4037
scDisabled enumeration member 4036
scDisabledSelected enumeration member 4036
scHover enumeration member 4036
scHoverSelected enumeration member 4036
scNormal enumeration member 4036
scPressed enumeration member 4036
ScreenTip
TApplicationMenu class 3355
ScreenTips
TCustomRibbon class 3398
TRibbon class 3496
ScriptContent
TInetXPageElements class 2284
ScriptEngine
TInetXPageProducer class 2290
ScriptManager
TBaseInetXPageProducer class 2272
Scroll
TOleControl class 2676
ScrollBars
TOutline class 2901
ScrollButtons
TCustomTabData class 3458
ScrollMode
TCustomRibbonQuickAccessToolbar class 3453
TRibbonQATScrollButton class 3771
scSelected enumeration member 4036
SecondaryShortCuts
TStaticListAction class 1060
TVirtualListAction class 1070
SectionExists
TCustomIniFile class 736
SecurityAction
TIBSecurityService class 450
SecurityDatabaseLocation
TConfigParams class 404
Seek
TFileStream class 2995
TIBBlobStream class 48
TIBDSBlobStream class 141
TResourceStream class 3004
TStream class 3007
SelectApplicationButton
TRibbonApplicationMenuBar class 3948
SelectedItem
TCustomOutline class 2872
SelectFirstItem
TRibbonActionPopupMenu class 3942
SelectNext
TRibbonTabs class 3563
SelectPrior
TRibbonTabs class 3563

SelectSQL
TIBCustomDataSet class 105
TIBDataSet class 131
SelectStmtHandle
TIBCustomDataSet class 105
TIBDataSet class 131
Send
ISOAPHeaders interface 926
TSOAPHeaders class 915
SendAt
ISOAPHeaders interface 926
TSOAPHeaders class 915
SendCount
ISOAPHeaders interface 927
TSOAPHeaders class 916
Sender
TChangeLink class 650
SendMisc
IIBSQLMonitorHook interface 570
SendRedirect
TISAPIResponse class 982
SendResponse
TISAPIResponse class 982
SendStream
TISAPIResponse class 982
Sent
TISAPIResponse class 983
SeparatorOpacity
TCustomTabData class 3458
SerializationOptions
TRemotable class 873
TSOAPDomConv class 2778
SerializeOptions
TRemotableTypeRegistry class 887
Server
TServerEventDispatch class 2724
ServerData
TOleServer class 2719
TSimpleObjectBroker class 2481
ServerDisconnect
TServerEventDispatch class 2723
ServerGUID
TDispatchConnection class 1301
ServerImplementation
TVersionInfo class 476
ServerName
TDispatchConnection class 1301
TIBCustomService class 431
ServerPort
TMidasConnection class 1786
Servers
TSimpleObjectBroker class 2481
ServerVersion
TVersionInfo class 476
ServiceAttach
IIBSQLMonitorHook interface 570
ServiceDetach
IIBSQLMonitorHook interface 570
ServiceParamBySPB
TIBCustomService class 432
ServiceQuery
IIBSQLMonitorHook interface 571
TOleControl class 2698
ServiceQueryParams
TIBCustomService class 432
ServiceStart
IIBSQLMonitorHook interface 571
TIBConfigService class 414
TIBControlService class 423
ServiceStartAddParam
TIBControlService class 424
ServiceStartParams
TIBControlService class 425
ServiceVersion
TVersionInfo class 476
Set_Value
ISharedProperty interface 2428
SetAbort

IObjectContext interface 2422
TMtsAutoObject class 2393
TMtsDataModule class 2409
SetAction
TListActionLink class 1040
SetActionClient
TRibbonComboControl class 3694
TRibbonCustomControl class 3705
SetActive
TIBCustomDataSet class 89
TListActionLink class 1040
SetActiveObject
TOleControl class 2677
SetActiveUpdateException
TCustomProvider class 3114
SetAsString
TIBStringField class 148
SetAsyncMode
TIBConfigService class 415
SetAttachments
IOPConvert interface 2736
SetAutoCheck
TMenuActionLink class 1359
SetAutoConnect
TOleServer class 2716
SetAutoRegister
TIBEvents class 262
SetAutoSizing
TCustomRibbonGroup class 3418
SetBlobID
TIBBlobStream class 48
SetBookmarkData
TIBCustomDataSet class 90
SetBookmarkFlag
TIBCustomDataSet class 90
SetBorderSpace
TOleControl class 2677
SetBounds
TCustomRibbon class 3389
TCustomRibbonComboBox class 3589
TCustomRibbonGroup class 3418
TCustomRibbonPage class 3435
TOleControl class 2677
TRibbonApplicationMenuBar class 3948
TRibbonComboControl class 3694
TRibbonCustomControl class 3705
TRibbonGalleryControlMenuItem class 3725
TRibbonSpinButton class 3800
SetBreak
TMenuItem class 1389
SetByteProp
TOleControl class 2677
SetCachedUpdates
TIBCustomDataSet class 90
SetCaption
TListControlItem class 1045
TMenuActionLink class 1359
TMenuItem class 1389
TWebButton class 2061
TWebDataDisplay class 2082
TWebDataInput class 2092
SetCaptionAttributes
TWebDataDisplay class 2083
SetChecked
TMenuActionLink class 1360
TMenuItem class 1390
SetChildOrder
TBasePageItemsProducer class 2969
TMenu class 1342
TMenuItem class 1390
SetColorMap
TCustomRibbonGalleryPopupBar class 3990
TRibbonApplicationPopupMenu class 3967
SetColorProp
TOleControl class 2678
SetCommandText
TDataSetProvider class 3158
SetComplete

IObjectContext interface 2423
TMtsAutoObject class 2393
TMtsDataModule class 2410
SetCompProp
TOleControl class 2678
SetConnected
TCOMConnection class 1280
TDispatchConnection class 1295
SetConnectKind
TOleServer class 2717
SetConnectStatus
TCustomObjectBroker class 1284
TSimpleObjectBroker class 2479
SetConnectType
TMidasConnection class 1785
SetContainedControl
IRibbonContainedControl interface 3915
TRibbonComboControl class 3694
TRibbonCustomControl class 3708
TRibbonGalleryControl class 3721
TRibbonGalleryControlMenuItem class 3727
SetContent
TISAPIResponse class 983
SetCount
TIBXSQLDA class 542
SetCurrencyProp
TOleControl class 2679
SetCurrentKey
TRegistry class 3280
SetCursor
TCustomMaskEdit class 1090
SetData
TListControlItem class 1045
SetDatabase
TIBBase class 172
TIBBlobStream class 48
TIBSQL class 528
SetDataPointer
TDataContext class 822
SetDataSet
TIBDataSetUpdateObject class 137
TIBUpdateSQL class 634
TWebListInput class 2119
SetDataSetField
TXMLDataSet class 2169
TXMLDisplayGroup class 2184
SetDataSource
TIBCustomDataSet class 90
SetDateVariable
TISAPIResponse class 983
SetDBSqlDialect
TIBConfigService class 415
SetDefault
TMenuItem class 1390
SetDefaultWebComponents
TWebControlGroup class 2071
TWebForm class 2110
SetDisplayComponent
TWebDisplay class 2097
TXMLDisplayReferenceButton class 2195
SetDisplayComponentParent
TXMLDisplayReferenceButton class 2195
SetDisplayWidth
TCustomOutline class 2860
SetDoubleProp
TOleControl class 2679
SetDragMode
TRibbonComboControl class 3695
TRibbonCustomControl class 3708
SetDSWriter
TCustomPacketWriter class 3096
SetEnabled
IIBSQLMonitorHook interface 571
TMenuActionLink class 1360
TMenuItem class 1390
TRibbonSpinEdit class 3843
SetEnableValidateFields
IValidateFields interface 2225

TXMLDisplayGroup class 2185
SetEncoding
IOPConvert interface 2737
SetFieldData
TIBCustomDataSet class 91
TPacketDataSet class 3194
SetFieldName
IDataSetField interface 2205
TWebDataInput class 2093
SetFiltered
TIBDataSet class 120
TIBQuery class 386
TIBStoredProc class 591
TIBTable class 613
SetFilterOptions
TIBTable class 613
SetFilterText
TIBTable class 614
SetGoodIndex
TCustomOutline class 2860
SetGroupIndex
TMenuItem class 1391
SetHandle
TIBDataBase class 188
TIBTransaction class 214
SetHeadersInOut
IHeadersSetter interface 920
TSOAPHeadersBase class 918
SetHeadingAttributes
TCustomDataGrid class 1816
SetHeight
TJPEGImage class 1006
SetHelpContext
TMenuActionLink class 1361
SetHint
TMenuActionLink class 1361
SetIDispatchProp
TOleControl class 2679
SetImageIndex
TListControlItem class 1046
TMenuActionLink class 1361
TMenuItem class 1391
SetImages
TListActionLink class 1040
SetIncludePathURL
TBaseInetXPageProducer class 2269
SetIntegerProp
TOleControl class 2680
SetIntegerVariable
TISAPIResponse class 984
SetItemIndex
TListActionLink class 1041
SetItems
TWebListInput class 2119
SetIUnknownProp
TOleControl class 2680
SetKeyTip
IRibbonKeyTip interface 3921
TCustomRibbonGroup class 3418
TRibbonApplicationMenuButton class 3610
TRibbonBaseButtonControl class 3625
TRibbonComboControl class 3695
TRibbonCustomControl class 3708
TRibbonGroupCollapsedControl class 3513
SetLevel
TCustomOutline class 2861
SetLines
TQueryTextArea class 2015
SetLogMessage
TISAPIResponse class 984
SetMenu
TOleControl class 2680
SetMenuIndex
TMenuItem class 1391
SetMethodInfo
TInvContext class 832
SetMode
TIBBlobStream class 49

SetName
TOleControl class 2681
SetNodeType
TSOAPDomConv class 2778
SetObjectPropFromText
TSOAPDomConv class 2778
SetOle2MenuHandle
TMainMenu class 1326
SetOleBoolProp
TOleControl class 2681
SetOleDateProp
TOleControl class 2681
SetOleEnumProp
TOleControl class 2682
SetOleVariantProp
TOleControl class 2682
SetOnExecute
TMenuActionLink class 1362
SetOptions
IOPConvert interface 2737
TSOAPDomConv class 2779
SetOwnership
TSOAPAttachment class 902
SetOwnsObjects
THeaderList class 827
SetOwnsSentHeaders
ISOAPHeaders interface 927
TSOAPHeaders class 916
SetPageBuffers
TIBConfigService class 415
SetPalette
TJPEGImage class 1006
SetPanCursor
TPanningWindow class 698
SetParamName
IQueryField interface 2218
TWebDataInput class 2093
SetParamPointer
TInvContext class 833
SetParams
TDataSetProvider class 3158
TIBUpdateSQL class 634
SetParent
TCustomRibbon class 3389
TCustomRibbonComboBox class 3589
TCustomRibbonGroup class 3419
TCustomRibbonPage class 3435
TCustomRibbonQuickAccessToolbar class 3450
TOffice2007Button class 3466
TOleControl class 2682
TRibbonApplicationMenuButton class 3610
TRibbonComboControl class 3695
TRibbonCustomControl class 3709
TRibbonGalleryControlMenuItem class 3727
TRibbonSpinEdit class 3843
SetParentComponent
TMenuItem class 1391
SetPopupPoint
TPopupMenu class 1434
SetPosition
TStream class 3007
SetPrepare
TIBStoredProc class 591
SetPrepared
TIBStoredProc class 591
SetPrincipal
IStub interface 2815
SetPrinter
TPrinter class 3050
SetProcName
TIBStoredProc class 592
SetPropDisplayString
TOleControl class 2683
SetProperty
TOleControl class 2683
SetRadioItem
TMenuItem class 1392
SetReadOnly

TIBConfigService class 416
SetRecNo
TIBCustomDataSet class 91
SetReserveSpace
TIBConfigService class 416
SetResultPointer
TInvContext class 833
SetRibbon
IRibbon interface 3567
TRibbonApplicationMenuBar class 3949
TRibbonApplicationPopupMenu class 3968
SetRowAttributes
TCustomDataGrid class 1816
SetSel
TCustomMaskEdit class 1090
SetServiceStartOptions
TIBBackupService class 410
TIBControlService class 424
TIBLicensingService class 436
TIBLogService class 439
TIBRestoreService class 441
TIBSecurityService class 448
TIBStatisticalService class 460
TIBValidationService class 464
SetShortCut
TMenuActionLink class 1362
TMenuItem class 1392
SetShortIntProp
TOleControl class 2684
SetSingleProp
TOleControl class 2684
SetSize
TCustomImageList class 674
TFileStream class 2995
TIBBlobStream class 49
TIBDSBlobStream class 141
TPNGPointerList class 3001
TResourceStream class 3004
TStream class 3007
SetSmallintProp
TOleControl class 2684
SetSourceFile
TSOAPAttachment class 902
SetSourceStream
TSOAPAttachment class 903
SetSourceString
TSOAPAttachment class 903
SetSQL
TIBSQL class 528
SetStateFieldValue
TPacketDataSet class 3194
SetStatusCode
TISAPIResponse class 984
SetStatusText
TOleControl class 2685
SetString
TCustomListAction class 1023
TCustomStaticListAction class 1031
SetStringProp
TOleControl class 2685
SetStrings
TMemIniFile class 778
SetStringVariable
TISAPIResponse class 985
SetStyles
TBaseInetXPageProducer class 2269
SetStylesFile
TBaseInetXPageProducer class 2270
SetSweepInterval
TIBConfigService class 416
SetTableAttributes
TCustomDataGrid class 1817
SetTColorProp
TOleControl class 2685
SetTDateTimeProp
TOleControl class 2686
SetTempDir
IOPConvert interface 2737

SetText
IQueryField interface 2219
TQueryRadioGroup class 1980
TQuerySelectOptions class 1993
TQueryText class 2004
TQueryTextArea class 2015
SetTFontProp
TOleControl class 2686
SetTOleEnumProp
TOleControl class 2686
SetTPictureProp
TOleControl class 2687
SetTraceFlags
IIBSQLMonitorHook interface 571
SetTransaction
TIBBase class 173
TIBBlobStream class 49
TIBSQL class 529
SetUpdateState
TCustomOutline class 2861
SetupDlg
TRibbonCustomizeDlg class 3499
SetupDropDownBtn
TCustomRibbonGroup class 3419
TCustomRibbonQuickAccessToolbar class 3451
SetupGroupImage
TRibbonGroupCollapsedControl class 3514
SetUseBroker
TMidasConnection class 1785
SetUseParent
TXMLDataParent class 2163
TXMLDataSetParent class 2175
TXMLDisplayParent class 2190
SetValues
TWebListInput class 2119
SetVariantProp
TOleControl class 2687
SetVisible
TMenuActionLink class 1362
TMenuItem class 1392
SetWebContentOptions
TBaseInetXPageProducer class 2270
SetWebDisplayComponent
TWebDisplayReferenceGroup class 2101
SetWideStringProp
TOleControl class 2688
SetWidth
TJPEGImage class 1006
SetWordBoolProp
TOleControl class 2688
SetWordProp
TOleControl class 2688
SetXMLBroker
TXMLButton class 2155
TXMLData class 2160
TXMLDataSet class 2169
TXMLDisplayGroup class 2185
SetXMLComponent
TQueryButton class 1959
SetXMLUseParent
TQueryButton class 1959
TXMLButton class 2155
sfBottomFrame enumeration member 4038
sfBottomFrameDisabled enumeration member 4038
sfButton enumeration member 4038
sfButtonHover enumeration member 4038
sfButtonPressed enumeration member 4038
sfClose enumeration member 4038
sfCloseDisabled enumeration member 4038
sfCloseHover enumeration member 4038
sfClosePressed enumeration member 4038
sfHelp enumeration member 4038
sfHelpDisabled enumeration member 4038
sfHelpHover enumeration member 4038
sfHelpPressed enumeration member 4038
sfLeftFrame enumeration member 4038
sfLeftFrameDisabled enumeration member 4038
sfLeftFrameTop enumeration member 4038

sfLeftFrameTopDisabled enumeration member 4038
sfMaximize enumeration member 4038
sfMaximizeDisabled enumeration member 4038
sfMaximizeHover enumeration member 4038
sfMaximizePressed enumeration member 4038
sfMinimize enumeration member 4038
sfMinimizeDisabled enumeration member 4038
sfMinimizeHover enumeration member 4038
sfMinimizePressed enumeration member 4038
sfNone enumeration member 4038
sfRestore enumeration member 4038
sfRestoreDisabled enumeration member 4038
sfRestoreHover enumeration member 4038
sfRestorePressed enumeration member 4038
sfRightFrame enumeration member 4038
sfRightFrameDisabled enumeration member 4038
sfRightFrameTop enumeration member 4038
sfRightFrameTopDisabled enumeration member 4038
sgButton enumeration member 4040
sgDownButton enumeration member 4040
sgDownButtonDisabled enumeration member 4040
sgDownButtonHover enumeration member 4040
sgDownButtonPressed enumeration member 4040
sgMoreButton enumeration member 4040
sgMoreButtonHover enumeration member 4040
sgMoreButtonHoverDisabled enumeration member 4040
sgUpButton enumeration member 4040
sgUpButtonDisabled enumeration member 4040
sgUpButtonHover enumeration member 4040
sgUpButtonPressed enumeration member 4040
Shareable
    TMediaPlayer class 2358
ShareImages
    TCustomImageList class 683
ShortCut
    TMenuItem class 1415
    TStaticListAction class 1060
    TVirtualListAction class 1071
Show
    TApplicationMenu class 3353
    TQuickAccessToolbar class 3473
ShowAboutBox
    TOleControl class 2689
ShowBorder
    TCustomRibbonQuickAccessToolbar class 3453
ShowComboItems
    TRibbonComboControl class 3696
ShowControls
    TCustomRibbonGroup class 3419
ShowCustomizeMenu
    TCustomRibbonQuickAccessToolbar class 3451
ShowHelpButton
    TCustomRibbon class 3398
    TRibbon class 3496
ShowHint
    TMaskEdit class 1134
    TMediaPlayer class 2358
    TOleContainer class 2577
    TOutline class 2901
    TRibbonComboBox class 3642
    TRibbonSpinButton class 3815
    TRibbonSpinEdit class 3861
ShowIBMessage enumeration member 15
ShowItemCaptions
    TCustomRibbonGalleryBar class 3988
    TRibbonGalleryBar class 3992
ShowKeyTip
    IRibbonKeyTip interface 3921
    TCustomRibbonGroup class 3419
    TRibbonApplicationMenuButton class 3611
    TRibbonBaseButtonControl class 3625
    TRibbonComboControl class 3696
    TRibbonCustomControl class 3709
    TRibbonGroupCollapsedControl class 3514
ShowKeyTips
    TCustomRibbonPage class 3436
    TRibbonApplicationMenuBar class 3949
ShowMinimize

TQuickAccessToolbar class 3474
ShowMoreCommands
TQuickAccessToolbar class 3474
ShowMoreControls
TCustomRibbonQuickAccessToolbar class 3451
ShowObject
TOleControl class 2689
ShowPosition
TQuickAccessToolbar class 3474
ShowPropertyFrame
TOleControl class 2689
ShowSQLCode enumeration member 15
ShowSQLMessage enumeration member 15
ShowSystem
TIBExtract class 282
Shutdown
IORB interface 2809
ShutdownDatabase
TIBConfigService class 417
Size
TIBBCDField class 62
TIBXSQLVAR class 554
TStream class 3008
SizeMode
TOleContainer class 2578
Skip
IEnumNames interface 2418
SkipTitles
TIBInputDelimitedFile class 505
smArrowLeftDisabledIcon enumeration member 4041
smArrowLeftIcon enumeration member 4041
smArrowRightDisabledIcon enumeration member 4041
smArrowRightIcon enumeration member 4041
smAutomation enumeration member 2522
smAutoSize enumeration member 2621
smBackground enumeration member 4041
smBanner enumeration member 4041
smCenter enumeration member 2621
smCheck enumeration member 4041
smCheckDisabled enumeration member 4041
smCheckDisabledIcon enumeration member 4041
smCheckIcon enumeration member 4041
smClip enumeration member 2621
smDisabled enumeration member 4041
smHover enumeration member 4041
smNormal enumeration member 4041
Smoothing
TJPEGImage class 1010
smOptionButton enumeration member 4041
smOptionButtonDisabled enumeration member 4041
smOptionButtonHover enumeration member 4041
smRadio enumeration member 4041
smRadioDisabled enumeration member 4041
smRegServer enumeration member 2522
smResize enumeration member 4041
smResizeBoth enumeration member 4041
smResizeVertical enumeration member 4041
smRichContentDisabled enumeration member 4041
smRichContentHover enumeration member 4041
smScale enumeration member 2621
smScrollButton enumeration member 4041
smScrollButtonDownIcon enumeration member 4041
smScrollButtonHover enumeration member 4041
smScrollButtonUpIcon enumeration member 4041
smSelected enumeration member 4041
smSeparator enumeration member 4041
smStandalone enumeration member 2522
smStretch enumeration member 2621
smUnregServer enumeration member 2522
SOAPHeaders
TRIO class 4064
SOAPToObject
TRemotable class 872
TRemotableXS class 894
TSOAPAttachment class 903
TSOAPHeader class 911
soCacheMimeResponse enumeration member 2740
soCustomFaultAtDetailsNode enumeration member 2740

soDocument enumeration member 2740
soDontClearOutBoundHeaders enumeration member 2740
soDontSendEmptyNodes enumeration member 2740
soLiteralParams enumeration member 2740
soReturnSuccessForFault enumeration member 2740
soRootRefNodesToBody enumeration member 2740
Sort
TListControlItems class 1051
SortType
TListControlItems class 1053
soSendMultiRefArray enumeration member 2740
soSendMultiRefObj enumeration member 2740
soSendUntyped enumeration member 2740
soTryAllSchema enumeration member 2740
Source
EOleException class 2499
TIBBatchMove class 36
TUpdateTree class 3212
SourceDirectory
TIBInstall class 316
SourceDoc
TOleContainer class 2579
SourceStream
TSOAPAttachment class 907
SourceString
TSOAPAttachment class 908
soUTF8EncodeXML enumeration member 2740
soUTF8InHeader enumeration member 2740
soXXXXHdr enumeration member 2740
Spacing
TRibbonApplicationMenuBar class 3956
SPX enumeration member 479
sqBackground enumeration member 4043
sqButton enumeration member 4043
sqButtonCustomizeIcon enumeration member 4043
sqButtonHover enumeration member 4043
sqButtonMoreIcon enumeration member 4043
sqButtonPressed enumeration member 4043
sqEnd enumeration member 4043
sqEndRTL enumeration member 4043
sqGroup enumeration member 4043
sqGroupHover enumeration member 4043
sqGroupPressed enumeration member 4043
SQL
TIBQuery class 392
TIBSQL class 534
TIBUpdateSQL class 637
SQLChanged
TIBUpdateSQL class 634
SQLChanging
TIBSQL class 529
SQLCode
EIBError class 5
SQLCommit enumeration member 556
SQLDDL enumeration member 556
SQLDelete enumeration member 556
SQLDialect
TIBDataBase class 196
SQLException
SQLException class 989
SQLException class 988
about SQLException class 988
ErrorCode 989
SQLException 989
SQLException constructor 989
SQLException Members 988
SQLException properties 989
SQLState 990
SQLExecProcedure enumeration member 556
SQLExecute
IIBSQLMonitorHook interface 572
SQLFetch
IIBSQLMonitorHook interface 572
SQLGetSegment enumeration member 556
SQLInsert enumeration member 556
SQLObjectCount
TIBDataBase class 197
TIBTransaction class 218

SQLObjects
TIBDataBase class 197
TIBTransaction class 218
SQLParams
TIBCustomDataSet class 106
SQLPrepare
IIBSQLMonitorHook interface 572
SQLPutSegment enumeration member 556
SQlRole
TIBSecurityService class 450
SQLRollback enumeration member 556
SQLSelect enumeration member 556
SQLSelectForUpdate enumeration member 556
SQLSetGenerator enumeration member 556
SQLStartTransaction enumeration member 556
SQLState
SQLException class 990
SQLType
TIBCustomDataSet class 91
TIBSQL class 534
TIBXSQLVAR class 554
SQLUnknown enumeration member 556
SQLUpdate enumeration member 556
sqMiddle enumeration member 4043
sqStart enumeration member 4043
sqStartRTL enumeration member 4043
srAbort enumeration member 332
srBackground enumeration member 4045
srBackgroundDisabled enumeration member 4045
srBody enumeration member 4045
srContinue enumeration member 332
srgBackground enumeration member 4046
srgBackgroundDisabled enumeration member 4046
srgBackgroundHover enumeration member 4046
srgCaption enumeration member 4046
srgCaptionDisabled enumeration member 4046
srgCaptionHover enumeration member 4046
srgDialogButton enumeration member 4046
srgDialogButtonHover enumeration member 4046
srgDialogButtonIcon enumeration member 4046
srgDialogButtonIconHover enumeration member 4046
srgDialogButtonIconPressed enumeration member 4046
srgDialogButtonPressed enumeration member 4046
srgMinimizeButton enumeration member 4046
srgMinimizeButtonHover enumeration member 4046
srgMinimizeButtonIcon enumeration member 4046
srgMinimizeButtonPressed enumeration member 4046
srgMinimizeButtonPressedIcon enumeration member 4046
srHeader enumeration member 4045
srHelp enumeration member 4045
srPage enumeration member 4045
ssbBar enumeration member 4048
ssbBarHover enumeration member 4048
ssbBarIcon enumeration member 4048
ssbBarPressed enumeration member 4048
ssbButton enumeration member 4048
ssbButtonActive enumeration member 4048
ssbButtonDisabled enumeration member 4048
ssbButtonDownIcon enumeration member 4048
ssbButtonHover enumeration member 4048
ssbButtonPressed enumeration member 4048
ssbButtonUpIcon enumeration member 4048
ssbrBackground enumeration member 4050
ssbrControl enumeration member 4050
ssbrGripper enumeration member 4050
ssbrNotification enumeration member 4050
ssButton enumeration member 4049
ssButtonDisabled enumeration member 4049
ssButtonDown enumeration member 4049
ssButtonDownIcon enumeration member 4049
ssButtonHover enumeration member 4049
ssButtonInactive enumeration member 4049
ssButtonPressed enumeration member 4049
ssButtonUp enumeration member 4049
ssButtonUpIcon enumeration member 4049
StandardEvent
TOleControl class 2690
Start

TMediaPlayer class 2360
StartButtons
TFormLayout class 1920
StartFields
TFormLayout class 1921
StartMode
TAutomation class 2507
StartPanning
TPanningWindow class 698
StartPos
TMediaPlayer class 2360
StartRecording
TMediaPlayer class 2329
StartRow
TFormLayout class 1921
StartTable
TFormLayout class 1921
StartTransaction
TIBTransaction class 214
State
TLimboTransactionInfo class 472
TOleContainer class 2579
StatementType
TIBCustomDataSet class 106
TIBDataSet class 132
TIBQuery class 392
StatusContext
TIBSetup class 321
StatusInternal
TIBSetup class 319
stBackground enumeration member 4051
stBoth enumeration member 1074
stData enumeration member 1074
Step
TMediaPlayer class 2330
stHover enumeration member 4051
stHoverSelected enumeration member 4051
stHoverUnselected enumeration member 4051
stKeyTipsActive enumeration member 4051
StmtHandle
TIBQuery class 392
TIBStoredProc class 594
stNone enumeration member 1074
stNormal enumeration member 4051
Stop
TMediaPlayer class 2330
StopPanning
TPanningWindow class 698
StorageInterface
TOleContainer class 2579
StoreDefs
TIBTable class 621
StoredProcedureNames
TIBStoredProc class 594
StoredProcName
TIBStoredProc class 595
StreamMetaData
TPacketDataSet class 3195
Strings
TCustomListAction class 1026
StringToObject
IORB interface 2809
StringToValue
TEnumPropDesc class 2634
StrToClear
TDataContext class 817
stScrollGroupLeft enumeration member 4051
stScrollGroupLeftHover enumeration member 4051
stScrollGroupRight enumeration member 4051
stScrollGroupRightHover enumeration member 4051
stScrollLeft enumeration member 4051
stScrollLeftHover enumeration member 4051
stScrollLeftPressed enumeration member 4051
stScrollRight enumeration member 4051
stScrollRightHover enumeration member 4051
stScrollRightPressed enumeration member 4051
stSeparator enumeration member 4051
stText enumeration member 1074

Style
TApplyUpdatesButton class 1802
TCaptionAttributes class 1806
TCustomDataNavigator class 1823
TCustomFieldGroup class 1829
TCustomLayoutGroup class 1835
TCustomOutline class 2872
TCustomQueryButtons class 1840
TCustomRibbon class 3398
TDataForm class 1849
TDataNavigator class 1858
TDataSetButton class 1865
TFieldGroup class 1873
TFieldRadioGroup class 1882
TFieldSelectOptions class 1891
TFieldStatus class 1897
TFieldText class 1904
TFieldTextArea class 1911
TGridAttributes class 1928
TGridRowAttributes class 1934
TLayoutGroup class 1942
TOutline class 2902
TQueryButton class 1960
TQueryButtons class 1964
TQueryFieldGroup class 1971
TQueryForm class 1975
TQueryRadioGroup class 1986
TQuerySelectOptions class 1998
TQueryText class 2009
TQueryTextArea class 2020
TRibbon class 3497
TSelectOptionsColumn class 2031
TStatusColumn class 2037
TTextAreaColumn class 2047
TTextColumn class 2054
TWebButton class 2063
TWebDataDisplay class 2085
TWebForm class 2111
TWebTextInput class 2150
StyleChanged
IRibbonGroup interface 3569
TCustomRibbonGroup class 3420
StyleRule
TApplyUpdatesButton class 1802
TCaptionAttributes class 1807
TCustomDataNavigator class 1824
TCustomFieldGroup class 1830
TCustomLayoutGroup class 1835
TCustomQueryButtons class 1840
TDataForm class 1850
TDataNavigator class 1858
TDataSetButton class 1865
TFieldGroup class 1874
TFieldRadioGroup class 1883
TFieldSelectOptions class 1892
TFieldStatus class 1898
TFieldText class 1904
TFieldTextArea class 1911
TGridAttributes class 1929
TGridRowAttributes class 1934
TLayoutGroup class 1942
TQueryButton class 1961
TQueryButtons class 1965
TQueryFieldGroup class 1971
TQueryForm class 1975
TQueryRadioGroup class 1986
TQuerySelectOptions class 1998
TQueryText class 2009
TQueryTextArea class 2020
TSelectOptionsColumn class 2032
TStatusColumn class 2038
TTextAreaColumn class 2047
TTextColumn class 2055
TWebButton class 2063
TWebDataDisplay class 2086
TWebForm class 2111
TWebTextInput class 2151
Styles

TBaseInetXPageProducer class 2272
TInetXPageProducer class 2291
StylesContent
TInetXPageElements class 2284
StylesFile
TBaseInetXPageProducer class 2273
TInetXPageProducer class 2291
SubMenuImages
TMenuItem class 1415
SuggestedDestination
TIBInstall class 316
SuppressException
TOleControl class 2690
SweepDB enumeration member 485
SweepInterval
TIBDatabaseInfo class 254
SystemRelations enumeration member 482

T

TabExists
TRibbonTabs class 3564
TabHandledKeyTips
TCustomRibbon class 3390
TabHideKeyTips
TCustomRibbon class 3390
TabIndex
TCustomRibbon class 3399
TFieldRadioGroup class 1883
TFieldSelectOptions class 1892
TFieldText class 1905
TFieldTextArea class 1912
TQueryRadioGroup class 1987
TQuerySelectOptions class 1999
TQueryText class 2010
TQueryTextArea class 2021
TRibbon class 3497
TSelectOptionsColumn class 2032
TTextAreaColumn class 2048
TTextColumn class 2055
TWebDataDisplay class 2086
TableAttributes
TCustomDataGrid class 1818
TDataGrid class 1855
TableHeader
TFormLayout class 1922
TableName
TIBTable class 621
TableNames
TIBTable class 622
TableTypes
TIBTable class 622
TabOrder
TMaskEdit class 1135
TMediaPlayer class 2361
TOleContainer class 2580
TOutline class 2903
TRibbonComboBox class 3644
TRibbonSpinButton class 3817
TRibbonSpinEdit class 3863
Tabs
TCustomRibbon class 3399
TRibbon class 3497
TabSeparatorState
TCustomTabData class 3459
TabShowKeyTips
TCustomRibbon class 3390
TabStop
TMaskEdit class 1136
TMediaPlayer class 2361
TOleContainer class 2580
TOleControl class 2697
TOutline class 2904
TRibbonComboBox class 3644
TRibbonSpinButton class 3818
TRibbonSpinEdit class 3864
TabUpdateRect
TCustomTabData class 3459
TACommit enumeration member 226

TACommitRetaining enumeration member 226
Tag
TOptionItem class 3937
TAlphaBitmap class 4017
about TAlphaBitmap class 4017
Create 4018
TAlphaBitmap Members 4017
TApplicationMenu class 3350
about TApplicationMenu class 3350
Assign 3352
Caption 3353
CommandType 3354
Create 3351
Destroy 3351
GetActionClientItem 3352
Icon 3354
IconSize 3354
Initialize 3353
KeyTip 3355
Menu 3355
ScreenTip 3355
Show 3353
TApplicationMenu Members 3350
TApplicationMenu methods 3351
TApplicationMenu properties 3353
TApplyUpdatesButton class 1798
about TApplyUpdatesButton class 1798
AddElements 1800
Caption 1801
Create 1799
Custom 1802
GetSubComponents 1800
ImplAddElements 1800
ImplContent 1800
Style 1802
StyleRule 1802
TApplyUpdatesButton Members 1799
TApplyUpdatesButton methods 1800
TApplyUpdatesButton properties 1801
XMLBroker 1803
XMLUseParent 1803
TARollback enumeration member 226
TARollbackRetaining enumeration member 226
TAutoDispatch class 2500
about TAutoDispatch class 2500
AddRef 2502
AutoObject 2504
Create 2501
GetAutoObject 2502
GetIDsOfNames 2502
GetTypeInfo 2502
GetTypeInfoCount 2502
Invoke 2503
QueryInterface 2503
Release 2503
TAutoDispatch Members 2501
TAutoDispatch methods 2501
TAutoDispatch properties 2504
TAutomation class 2504
about TAutomation class 2504
AutoObjectCount 2506
Create 2505
Destroy 2505
IsInprocServer 2506
OnLastRelease 2507
RegisterClass 2505
StartMode 2507
TAutomation events 2507
TAutomation Members 2504
TAutomation methods 2505
TAutomation properties 2506
UpdateRegistry 2506
TAutoObject class 2507
about TAutoObject class 2507
AddRef 2509
AutoDispatch 2510
Create 2508
CreateAutoDispatch 2510

Destroy 2508
GetExceptionInfo 2510
OleObject 2511
RefCount 2511
Release 2510
TAutoObject Members 2508
TAutoObject methods 2509
TAutoObject properties 2510
TBaseInetXPageProducer class 2257
about TBaseInetXPageProducer class 2257
AddIncludes 2261
AddScriptComponents 2261
CanAddClass 2261
ContentFromStream 2262
Create 2259
CreatePageElements 2262
CreateScriptManager 2262
DeclareFunctions 2263
DeclareIncludes 2263
DeclareRowSets 2263
DeclareScript 2263
DeclareStyles 2263
DeclareWarnings 2264
DeclareXMLDocuments 2264
DeclareXMLIslands 2264
Destroy 2259
DoAfterGetContent 2264
DoAfterGetXMLData 2265
DoBeforeGetContent 2265
DoBeforeGetXMLData 2265
DoTagEvent 2265
EnableXMLIslands 2271
FindXMLBroker 2265
FindXMLBrokerName 2266
GeneratePageElements 2266
GetContentOptions 2267
GetDefaultTemplate 2267
GetRequiredTags 2267
GetScriptManager 2268
GetUseXMLIslands 2268
GetXMLData 2268
GetXMLDelta 2268
GetXMLErrors 2269
ImplCanAddClass 2269
IncludePathURL 2271
OnAfterGetContent 2273
OnAfterGetXMLData 2274
OnBeforeGetContent 2274
OnBeforeGetXMLData 2275
PageElements 2271
ScriptManager 2272
SetIncludePathURL 2269
SetStyles 2269
SetStylesFile 2270
SetWebContentOptions 2270
Styles 2272
StylesFile 2273
TBaseInetXPageProducer events 2273
TBaseInetXPageProducer Members 2258
TBaseInetXPageProducer methods 2260
TBaseInetXPageProducer properties 2270
TBasePageItemsProducer class 2966
about TBasePageItemsProducer class 2966
Create 2967
Destroy 2967
GetChildren 2968
GetComponentList 2968
GetDefaultComponentList 2969
SetChildOrder 2969
TBasePageItemsProducer Members 2967
TBasePageItemsProducer methods 2968
TBasePageItemsProducer properties 2969
WebPageItems 2969
TBaseProvider class 3077
about TBaseProvider class 3077
AfterUpdateRecord 3090
BeforeUpdateRecord 3090
CheckResolver 3081

Create 3079
CreateDataPacket 3081
CreateResolver 3082
Destroy 3080
DoAfterUpdateRecord 3082
DoBeforeCommit 3082
DoBeforeUpdateRecord 3083
DoOnGetData 3083
DoOnUpdateData 3083
FetchDetails 3083
FreeResolver 3084
InternalApplyUpdates 3084
InternalGetRecords 3085
InternalRowRequest 3086
LocateRecord 3086
OnDataRequest 3091
OnGetData 3091
OnUpdateData 3092
OnUpdateError 3092
Options 3087
PacketDataSet 3088
Resolver 3088
TBaseProvider events 3089
TBaseProvider Members 3078
TBaseProvider methods 3080
TBaseProvider properties 3087
UpdateMode 3089
UpdateRecord 3087
tbLeftButton enumeration member 1463
tbRightButton enumeration member 1463
TCaptionAttributes class 1804
about TCaptionAttributes class 1804
Align 1805
Create 1805
Custom 1806
Parent 1806
Style 1806
StyleRule 1807
TCaptionAttributes Members 1804
TCaptionAttributes properties 1805
VAlign 1807
TChangeLink class 648
about TChangeLink class 648
Change 649
Destroy 649
OnChange 650
Sender 650
TChangeLink events 650
TChangeLink Members 648
TChangeLink methods 649
TChangeLink properties 650
TCOMConnection class 1278
about TCOMConnection class 1278
DoConnect 1280
SetConnected 1280
TCOMConnection Members 1279
TCOMConnection methods 1279
TConfigFileData class 399
about TConfigFileData class 399
ConfigFileKey 401
ConfigFileValue 401
Create 400
Destroy 400
TConfigFileData fields 400
TConfigFileData Members 400
TConfigParams class 401
about TConfigParams class 401
BaseLocation 402
ConfigFileData 403
ConfigFileParams 403
Create 402
Destroy 402
LockFileLocation 403
MessageFileLocation 403
SecurityDatabaseLocation 404
TConfigParams fields 402
TConfigParams Members 401
TCP enumeration member 479

TCustomActionControlBar class 3355
about TCustomActionControlBar class 3355
CMControlChange 3357
Create 3356
CreateControl 3357
CreateControls 3357
CreatingControls 3358
GetControlClass 3357
TCustomActionControlBar Members 3356
TCustomActionControlBar methods 3356
TCustomActionControlBar properties 3357
TCustomDataForm class 1808
about TCustomDataForm class 1808
GetHTMLFormTag 1810
ImplCanAddClass 1810
ImplContent 1810
TCustomDataForm Members 1808
TCustomDataForm methods 1809
TCustomDataGrid class 1811
about TCustomDataGrid class 1811
Create 1814
Destroy 1814
DisplayRows 1817
FormatTable 1814
HeadingAttributes 1817
ImplAddElements 1815
ImplCanAddClass 1815
ImplContent 1815
ImplIsMultipleRecordView 1816
RowAttributes 1818
SetHeadingAttributes 1816
SetRowAttributes 1816
SetTableAttributes 1817
TableAttributes 1818
TCustomDataGrid Members 1812
TCustomDataGrid methods 1814
TCustomDataGrid properties 1817
TCustomDataNavigator class 1819
about TCustomDataNavigator class 1819
Custom 1823
GetDefaultButtons 1821
ImplCanAddClass 1821
ImplContent 1821
ImplGetSubComponents 1822
Style 1823
StyleRule 1824
TCustomDataNavigator Members 1820
TCustomDataNavigator methods 1821
TCustomDataNavigator properties 1823
VisibleButtons 1822
TCustomFieldGroup class 1824
about TCustomFieldGroup class 1824
Create 1827
Custom 1829
Destroy 1827
GetLayoutAttributes 1828
ImplContent 1828
Style 1829
StyleRule 1830
TCustomFieldGroup Members 1825
TCustomFieldGroup methods 1828
TCustomFieldGroup properties 1829
TCustomImageList class 651
about TCustomImageList class 651
Add 655
AddIcon 655
AddImage 656
AddImages 656
AddMasked 657
AllocBy 675
Assign 657
AssignTo 658
BkColor 676
BlendColor 676
Change 658
Clear 658
ColorDepth 677
Count 677

Create 653
CreateSize 653
DefineProperties 659
Delete 659
Destroy 654
DoDraw 659
Draw 660
Draw method 660
DrawingStyle 681
DrawOverlay 663
DrawOverlay method 663
FileLoad 664
GetBitmap 664
GetHotSpot 665
GetIcon 665
GetIcon method 665
GetImageBitmap 666
GetImages 666
GetInstRes 667
GetInstRes method 667
GetMaskBitmap 668
GetResource 668
Handle 682
HandleAllocated 669
HandleNeeded 669
Height 682
ImageType 682
Initialize 669
Insert 670
InsertIcon 670
InsertMasked 670
Masked 683
Move 671
OnChange 684
Overlay 671
ReadData 672
RegisterChanges 672
Replace 672
ReplaceIcon 673
ReplaceMasked 673
ResInstLoad 673
ResourceLoad 674
SetSize 674
ShareImages 683
TCustomImageList events 684
TCustomImageList Members 651
TCustomImageList methods 654
TCustomImageList properties 675
UnRegisterChanges 674
Width 684
WriteData 675
TCustomInetXPageProducer class 2275
about TCustomInetXPageProducer class 2275
Content 2279
Create 2278
Destroy 2278
GetTemplateFileName 2280
GetTemplateStream 2280
HTMLDoc 2281
HTMLFile 2282
TCustomInetXPageProducer Members 2276
TCustomInetXPageProducer methods 2279
TCustomInetXPageProducer properties 2281
WebContent 2281
TCustomIniFile class 716
about TCustomIniFile class 716
Create 720
DeleteKey 721
EraseSection 722
FileName 749
ReadBinaryStream 722
ReadBool 723
ReadDate 726
ReadDateTime 726
ReadFloat 727
ReadInteger 727
ReadSection 730
ReadSections 731

ReadSections method 731
ReadSectionValues 732
ReadString 732
ReadTime 735
SectionExists 736
TCustomIniFile Members 720
TCustomIniFile methods 721
TCustomIniFile properties 749
UpdateFile 736
ValueExists 736
WriteBinaryStream 737
WriteBool 737
WriteDate 741
WriteDateTime 741
WriteFloat 742
WriteInteger 742
WriteString 745
WriteTime 748
TCustomLayoutGroup class 1830
about TCustomLayoutGroup class 1830
Create 1832
Custom 1834
Destroy 1832
DisplayColumns 1834
GetLayoutAttributes 1832
ImplCanAddClass 1833
ImplContent 1833
Style 1835
StyleRule 1835
TCustomLayoutGroup Members 1831
TCustomLayoutGroup methods 1832
TCustomLayoutGroup properties 1834
TCustomListAction class 1019
about TCustomListAction class 1019
Active 1024
Count 1024
Create 1020
ExecuteTarget 1021
GetCount 1021
GetString 1022
HandlesTarget 1022
Images 1025
ItemIndex 1025
Loaded 1023
Loading 1026
OnGetItemCount 1027
OnItemSelected 1027
SetString 1023
Strings 1026
TCustomListAction events 1026
TCustomListAction Members 1020
TCustomListAction methods 1021
TCustomListAction properties 1023
TCustomMaskEdit class 1080
about TCustomMaskEdit class 1080
CheckCursor 1083
Clear 1083
Create 1082
EditCanModify 1084
EditMask 1092
EditText 1095
GetFirstEditChar 1084
GetLastEditChar 1085
GetMaxChars 1085
GetNextEditChar 1086
GetPriorEditChar 1086
GetSel 1087
GetTextLen 1087
IsMasked 1096
KeyDown 1087
KeyPress 1088
KeyUp 1088
MaskState 1096
MaxLength 1096
ReformatText 1089
Reset 1089
SetCursor 1090
SetSel 1090

TCustomMaskEdit Members 1081
TCustomMaskEdit methods 1082
TCustomMaskEdit properties 1092
Text 1097
Validate 1091
ValidateEdit 1091
ValidateError 1091
TCustomObjectBroker class 1281
about TCustomObjectBroker class 1281
GetComputerForGUID 1282
GetComputerForProgID 1283
GetPortForComputer 1283
SetConnectStatus 1284
TCustomObjectBroker Members 1282
TCustomObjectBroker methods 1282
TCustomOutline class 2838
about TCustomOutline class 2838
Add 2842
AddChild 2843
AddChildObject 2844
AddObject 2844
BadIndex 2845
BeginUpdate 2846
Canvas 2862
Clear 2846
Click 2846
Collapse 2847
Create 2840
DblClick 2847
DefineProperties 2847
Delete 2848
DeleteNode 2848
Destroy 2841
DrawCell 2849
EndUpdate 2849
Expand 2850
FullCollapse 2850
FullExpand 2850
GetDataItem 2851
GetItem 2851
GetNodeDisplayWidth 2852
GetTextItem 2852
GetVisibleNode 2852
Insert 2853
InsertObject 2853
InvalidateCell 2854
ItemCount 2864
ItemHeight 2864
Items 2865
ItemSeparator 2865
KeyDown 2855
KeyPress 2855
Lines 2866
Loaded 2856
LoadFromFile 2856
LoadFromFile method 2856
LoadFromStream 2857
LoadFromStream method 2857
MouseDown 2858
Move 2858
OnCollapse 2873
OnDrawItem 2873
OnExpand 2873
Options 2866
OutlineStyle 2867
PictureClosed 2868
PictureLeaf 2868
PictureMinus 2869
PictureOpen 2869
PicturePlus 2870
Row 2870
SaveToFile 2859
SaveToFile method 2858
SaveToStream 2859
SaveToStream method 2859
SelectedItem 2872
SetDisplayWidth 2860
SetGoodIndex 2860

SetLevel 2861
SetUpdateState 2861
Style 2872
TCustomOutline events 2872
TCustomOutline Members 2839
TCustomOutline methods 2841
TCustomOutline properties 2862
TCustomPacketWriter class 3093
about TCustomPacketWriter class 3093
AddAttribute 3095
Check 3095
Create 3094
Destroy 3094
DSWriter 3096
FBuffer 3095
SetDSWriter 3096
TCustomPacketWriter fields 3095
TCustomPacketWriter Members 3093
TCustomPacketWriter methods 3095
TCustomPacketWriter properties 3096
TCustomProvider class 3096
about TCustomProvider class 3096
ActiveUpdateException 3114
AfterApplyUpdates 3116
AfterExecute 3117
AfterGetParams 3118
AfterGetRecords 3118
AfterRowRequest 3119
ApplyUpdates 3100
ApplyUpdates method 3100
BeforeApplyUpdates 3120
BeforeExecute 3121
BeforeGetParams 3121
BeforeGetRecords 3122
BeforeRowRequest 3123
Create 3099
Data 3115
DataRequest 3101
Destroy 3099
DoAfterApplyUpdates 3102
DoAfterExecute 3102
DoAfterGetParams 3102
DoAfterGetRecords 3103
DoAfterRowRequest 3103
DoApplyUpdates 3104
DoBeforeApplyUpdates 3104
DoBeforeExecute 3105
DoBeforeGetParams 3105
DoBeforeGetRecords 3106
DoBeforeRowRequest 3106
DoGetRecords 3107
Execute 3107
Exported 3115
GetParams 3108
GetRecords 3109
GetRecords method 3109
InternalApplyUpdates 3110
InternalExecute 3111
InternalGetParams 3111
InternalGetRecords 3112
InternalRowRequest 3113
OnDataRequest 3124
OnValidate 3125
OwnerData 3115
RowRequest 3113
SetActiveUpdateException 3114
TCustomProvider events 3116
TCustomProvider Members 3097
TCustomProvider methods 3099
TCustomProvider properties 3114
TCustomQueryButtons class 1836
about TCustomQueryButtons class 1836
Custom 1839
GetDefaultButtons 1837
ImplAddElements 1838
ImplCanAddClass 1838
ImplContent 1838
Style 1840

StyleRule 1840
TCustomQueryButtons Members 1836
TCustomQueryButtons methods 1837
TCustomQueryButtons properties 1839
VisibleButtons 1839
TCustomQueryForm class 1841
about TCustomQueryForm class 1841
Action 1846
AssignStrings 1843
AssignStringsCallback 1844
Create 1843
GetHTMLFormTag 1844
ImplAddElements 1844
ImplCanAddClass 1845
ImplContent 1845
Method 1846
MethodString 1846
TCustomQueryForm Members 1842
TCustomQueryForm methods 1843
TCustomQueryForm properties 1846
TCustomResolver class 3125
about TCustomResolver class 3125
ApplyUpdates 3128
BeginUpdate 3128
Create 3126
Destroy 3127
DoDelete 3129
DoInsert 3129
DoUpdate 3129
EndUpdate 3130
FreeTreeData 3130
HandleUpdateError 3131
InitializeConflictBuffer 3131
InitKeyFields 3132
InitTreeData 3132
InternalBeforeResolve 3133
InternalUpdateRecord 3133
LogUpdateError 3133
LogUpdateRecord 3134
Provider 3135
RowRequest 3134
TCustomResolver Members 3126
TCustomResolver methods 3127
TCustomResolver properties 3135
UpdateTree 3135
TCustomRibbon class 3358
about TCustomRibbon class 3358
ActionManager 3394
Active 3394
ActivePage 3394
AddRecentItem 3370
AddTab 3370
AllowKeyboardNavigation 3395
ApplicationMenu 3395
BeginUpdate 3370
CheckRibbonSize 3370
ClearRecentItems 3371
CMBidimodechanged 3371
CMDesignHitTest 3371
CMFontchanged 3371
CMMouseenter 3371
CMMouseleave 3372
CMTextchanged 3372
CMThemeChanged 3372
ColorMap 3395
ContextMenuPopupHandler 3372
Create 3362
CreateContextMenu 3373
cRibbonFirstTabOffSet 3364
cRibbonGroupCaptionHeight 3364
cRibbonGroupHeight 3364
cRibbonHeight 3364
cRibbonHideHeight 3365
cRibbonHideWidth 3365
cRibbonMinimizedHeight 3365
cRibbonMinimumCaptionWidth 3365
cRibbonPageHeight 3366
cRibbonQATHeight 3366

cRibbonQuickAccessToolbarLeft 3366
cRibbonQuickAccessToolbarTop 3366
cRibbonTabHeight 3366
cRibbonTabScrollButtonWidth 3367
cRibbonTabSpacing 3367
cRibbonUnthemedCaptionHeight 3367
CustomFrameSupported 3373
CustomFrameWndProc 3373
DblClick 3373
DefineProperties 3374
DeleteTab 3374
Destroy 3363
DisplayKeyTips 3374
DoAddToQuickAccessToolbar 3375
DoChangeQuickAccessToolbarPosition 3375
DoContextPopup 3375
DocumentName 3396
DoCustomizeQuickAccessToolbar 3375
DoGroupStyleChanged 3376
DoHelpButtonClick 3376
DoMinimizeRibbon 3376
DoMouseWheelDown 3376
DoMouseWheelEx 3377
DoMouseWheelUp 3377
DoRecentItemClick 3377
DoScrollTabsLeft 3378
DoScrollTabsRight 3378
DoTabChange 3378
DrawCaption 3378
DrawQuickAccessToolbar 3379
DrawSystemButtons 3379
DrawTab 3379
EndUpdate 3379
GetContextMenuOptions 3379
GetHelpButtonRect 3380
GetIcon 3380
GetRibbonHeight 3380
GetRibbonMetric 3380
GetTabAt 3381
GetTabRect 3381
GetTabWidth 3381
HasApplicationMenu 3381
HasQuickAccessToolbar 3382
HideRibbon 3396
HideTabs 3396
HotTabIndex 3396
Idle 3382
InvalidateHeaderArea 3382
InvalidateScrollButtons 3382
InvalidateSystemMenuArea 3383
InvalidateTabArea 3383
KeyDown 3383
KeyTipsVisible 3397
Loaded 3384
Menu 3397
Minimized 3397
MouseDown 3384
MouseInSystemMenu 3385
MouseMove 3385
MouseUp 3385
MoveGroup 3386
Notification 3386
OnHelpButtonClick 3400
OnLeftTabScrollButtonClick 3400
OnRecentItemClick 3401
OnRightTabScrollButtonClick 3401
OnTabChange 3401
Paint 3387
QuickAccessToolbar 3397
RebuildRibbonMetrics 3387
RemoveAllTabs 3387
RemoveRecentItem 3388
RemoveTab 3388
Resize 3388
ScaleFactor 3398
ScreenTips 3398
SetBounds 3389
SetParent 3389

ShowHelpButton 3398
Style 3398
TabHandledKeyTips 3390
TabHideKeyTips 3390
TabIndex 3399
Tabs 3399
TabShowKeyTips 3390
TCustomRibbon constants 3363
TCustomRibbon events 3400
TCustomRibbon Members 3359
TCustomRibbon methods 3368
TCustomRibbon nested types 3399
TCustomRibbon properties 3393
TTabChangeEvent 3400
UM_CHECKSIZE 3367
UM_DISPLAYKEYTIPS 3368
UMCheckResize 3391
UMDisplayKeyTips 3391
UpdateQATPosition 3391
UpdateSystemMenuInfo 3391
UseCustomFrame 3399
ValidTab 3392
WMEraseBkgnd 3392
WMGetDlgCode 3392
WMKillFocus 3392
WMSettingChange 3393
WMWindowPosChanged 3393
TCustomRibbonColorMap class 3401
about TCustomRibbonColorMap class 3401
ActiveTabFontColor 3402
BevelMinimizedInner 3403
BevelMinimizedOuter 3403
CaptionFontColor 3403
ControlColor 3403
ControlFrameColor 3404
ControlInactiveColor 3404
DocumentFontColor 3404
GroupFontColor 3404
InactiveTabFontColor 3404
QuickAccessToolbarMoreBorderColor 3405
QuickAccessToolbarMoreColor 3405
TCustomRibbonColorMap Members 3402
TCustomRibbonColorMap properties 3402
TCustomRibbonComboBox class 3583
about TCustomRibbonComboBox class 3583
BeginAutoDrag 3585
CMMouseenter 3585
CMMouseleave 3586
Color 3591
Create 3584
Destroy 3584
DragDrop 3586
DragOver 3586
Items 3592
KeyDown 3587
KeyPress 3587
MouseInControl 3592
MouseMove 3588
Notification 3588
SetBounds 3589
SetParent 3589
TCustomRibbonComboBox Members 3583
TCustomRibbonComboBox methods 3584
TCustomRibbonComboBox properties 3590
WMKillFocus 3590
WMSetFocus 3590
TCustomRibbonGalleryBar class 3982
about TCustomRibbonGalleryBar class 3982
Align 3984
AlignControls 3983
Create 3983
CreateControl 3984
DoDropCategory 3984
ItemsPerRow 3988
ShowItemCaptions 3988
TCustomRibbonGalleryBar Members 3982
TCustomRibbonGalleryBar methods 3983
TCustomRibbonGalleryBar properties 3984

TCustomRibbonGalleryPopupBar class 3988
about TCustomRibbonGalleryPopupBar class 3988
AlignControls 3989
CreateControls 3990
DoGetControlClass 3990
SetColorMap 3990
TCustomRibbonGalleryPopupBar Members 3989
TCustomRibbonGalleryPopupBar methods 3989
TCustomRibbonGroup class 3405
about TCustomRibbonGroup class 3405
AlignControls 3409
AlignGroupControls 3410
Assign 3410
AutoSizingChanged 3410
BarHandleKeyTips 3410
BarHideKeyTips 3411
BarShowKeyTips 3411
CanCollapse 3421
Clone 3411
CMFontchanged 3412
CMItemDropPoint 3412
CMMouseenter 3412
CMMouseleave 3412
CMTextchanged 3413
CollapsedControl 3421
Create 3407
CreateControl 3413
CreateControls 3413
Destroy 3408
DialogAction 3421
DockDrop 3413
DoDropItem 3414
DoEndDrag 3414
DragDrop 3414
DrawShowDialogButton 3414
GetCaptionHeight 3414
GetGroupHeight 3415
GetKeyTip 3415
GroupAlign 3421
GroupIndex 3422
GroupVariant 3422
GroupWidth 3422
HandleKeyTip 3415
HasGallery 3423
HideControls 3416
HideKeyTip 3416
Items 3423
KeyTip 3423
KeyTipClick 3416
MouseDown 3417
MouseMove 3417
MouseUp 3417
Notification 3417
Page 3423
Paint 3418
RequiredWidth 3424
Resize 3418
Rows 3424
SetAutoSizing 3418
SetBounds 3418
SetKeyTip 3418
SetParent 3419
SetupDropDownBtn 3419
ShowControls 3419
ShowKeyTip 3419
StyleChanged 3420
TCustomRibbonGroup enumerations 3408
TCustomRibbonGroup Members 3406
TCustomRibbonGroup methods 3408
TCustomRibbonGroup nested types 3424
TCustomRibbonGroup properties 3420
TRowPosition 3424
TCustomRibbonPage class 3425
about TCustomRibbonPage class 3425
AddGroup 3429
AlignGroups 3429
CMControlListChanging 3429
CMTextchanged 3429

CollapseGroup 3430
CollapseGroupCount 3437
Create 3426
CreateScrollButton 3430
cRibbonGroupLeftOffset 3428
cRibbonGroupSpacing 3428
Destroy 3427
DoScrollButtonClick 3430
FindGroup 3430
FindGroupAt 3430
GetGroupPos 3431
GetNextVisibleGroup 3431
GetPageHeight 3431
GetPageTop 3431
GroupCount 3437
GroupOffset 3437
Groups 3437
HandleKeyTips 3432
HideKeyTips 3432
Index 3438
Invalidate 3432
KeyTip 3438
Loaded 3432
Notification 3433
Paint 3433
RequestAlign 3434
Resize 3434
Ribbon 3438
RibbonStyle 3434
SetBounds 3435
SetParent 3435
ShowKeyTips 3436
TCustomRibbonPage constants 3427
TCustomRibbonPage Members 3425
TCustomRibbonPage methods 3428
TCustomRibbonPage properties 3436
UpdateActionManager 3436
UpdateScrollButtons 3436
TCustomRibbonQuickAccessPopupToolbar class 3438
about TCustomRibbonQuickAccessPopupToolbar class 3438
Create 3439
CreateControl 3440
CreateWnd 3440
DrawBackground 3440
GetControlClass 3441
NCPaint 3441
TCustomRibbonQuickAccessPopupToolbar Members 3439
TCustomRibbonQuickAccessPopupToolbar methods 3439
ValidOrientations 3441
TCustomRibbonQuickAccessToolbar class 3441
about TCustomRibbonQuickAccessToolbar class 3441
BarHandleKeyTips 3444
BarHideKeyTips 3444
BarShowKeyTips 3445
CMFontchanged 3445
Create 3443
CreateControl 3445
CreateControls 3446
Customize 3446
DoContextPopup 3446
DoDropActions 3446
DoDropCategory 3447
DoDropItem 3447
DoMoreCommands 3447
DoScrollBtnClick 3447
DoUpdateQATWidth 3448
DragDrop 3448
DrawBackground 3448
GetBarHeight 3448
GetBarWidth 3449
GetScrollBtnClass 3449
Items 3452
Notification 3449
OnMoreCommands 3452
RequestAlign 3450
Ribbon 3452
ScrollMode 3453
SetParent 3450

SetupDropDownBtn 3451
ShowBorder 3453
ShowCustomizeMenu 3451
ShowMoreControls 3451
TCustomRibbonQuickAccessToolbar Members 3442
TCustomRibbonQuickAccessToolbar methods 3443
TCustomRibbonQuickAccessToolbar nested types 3453
TCustomRibbonQuickAccessToolbar properties 3451
TMoreCommandsEvent 3453
UpdateQATWidth 3451
TCustomRibbonSkin class 4018
about TCustomRibbonSkin class 4018
Bitmap 4022
Create 4019
Destroy 4019
DoLoadByFile 4021
DoLoadByResource 4021
GetLoadType 4021
GetLoadTypeValue 4021
GetResourceHandle 4021
LoadSkin 4022
TCustomRibbonSkin enumerations 4019
TCustomRibbonSkin Members 4018
TCustomRibbonSkin methods 4020
TCustomRibbonSkin properties 4022
TCustomStaticListAction class 1027
about TCustomStaticListAction class 1027
Clients 1032
Count 1032
Create 1029
Destroy 1029
GetCount 1030
GetItem 1030
GetItemClass 1030
GetString 1031
Items 1032
OnGetItem 1033
SetString 1031
TCustomStaticListAction events 1033
TCustomStaticListAction Members 1028
TCustomStaticListAction methods 1029
TCustomStaticListAction properties 1031
TCustomTabData class 3453
about TCustomTabData class 3453
Add 3455
AvailableSpace 3456
CalculateTabWidths 3455
Clear 3456
Create 3454
Destroy 3455
FirstVisibleTabIndex 3456
FullWidth 3457
ItemCount 3457
LastVisibleTabIndex 3457
LeftTabScrollButtonRect 3457
NewWidth 3458
Owner 3458
RightTabScrollButtonRect 3458
ScrollButtons 3458
SeparatorOpacity 3458
TabSeparatorState 3459
TabUpdateRect 3459
TCustomTabData Members 3454
TCustomTabData methods 3455
TCustomTabData properties 3456
TCustomVirtualListAction class 1034
about TCustomVirtualListAction class 1034
Count 1036
GetItem 1035
GetString 1035
OnGetItem 1037
TCustomVirtualListAction events 1037
TCustomVirtualListAction Members 1034
TCustomVirtualListAction methods 1035
TCustomVirtualListAction properties 1036
TDatabaseInfo class 404
about TDatabaseInfo class 404
Create 405

DbName 405
Destroy 405
NoOfAttachments 405
NoOfDatabases 406
TDatabaseInfo fields 405
TDatabaseInfo Members 404
TDataContext class 814
about TDataContext class 814
AddDynArrayToClear 818
AddObjectToDestroy 819
AddStrToClear 819
AddUStrToClear 820
AddVariantToClear 820
AddWStrToClear 821
AllocData 821
Create 815
Data 816
DataOffset 816
DataP 817
Destroy 815
DynArrayToClear 817
FObjsToDestroy 817
GetDataPointer 822
RemoveObjectToDestroy 822
SetDataPointer 822
StrToClear 817
TDataContext fields 816
TDataContext Members 815
TDataContext methods 818
UStrToClear 817
VarToClear 818
WStrToClear 818
TDataForm class 1847
about TDataForm class 1847
Custom 1849
Style 1849
StyleRule 1850
TDataForm Members 1848
TDataForm properties 1849
TDataGrid class 1850
about TDataGrid class 1850
DisplayRows 1853
HeadingAttributes 1854
RowAttributes 1854
TableAttributes 1855
TDataGrid Members 1851
TDataGrid properties 1853
TDataNavigator class 1855
about TDataNavigator class 1855
Custom 1857
Style 1858
StyleRule 1858
TDataNavigator Members 1856
TDataNavigator properties 1857
XMLComponent 1859
TDataPacketWriter class 3136
about TDataPacketWriter class 3136
AddColumn 3138
AddConstraints 3138
AddDataSetAttributes 3139
AddExtraFieldProps 3139
AddFieldLinks 3139
AddIndexDefs 3139
Constraints 3144
Destroy 3137
FreeInfoRecords 3140
GetDataPacket 3140
GetFieldIdx 3140
InitPutProcs 3140
OnGetParams 3144
Options 3144
PacketOptions 3144
PutADTField 3140
PutArrayField 3141
PutBlobField 3141
PutCalcField 3141
PutDataSetField 3141
PutField 3142

PutStringField 3142
PutVarBytesField 3142
PutWideStringField 3142
RefreshPutProcs 3142
Reset 3143
TDataPacketWriter events 3144
TDataPacketWriter Members 3136
TDataPacketWriter methods 3138
TDataPacketWriter properties 3143
WriteDataSet 3143
WriteMetaData 3143
TDataSetButton class 1859
about TDataSetButton class 1859
AddElements 1862
Caption 1864
Custom 1865
DefaultCaption 1862
GetCaption 1862
GetMethodName 1863
GetSubComponents 1863
ImplAddElements 1863
ImplContent 1863
Style 1865
StyleRule 1865
TDataSetButton fields 1861
TDataSetButton Members 1860
TDataSetButton methods 1862
TDataSetButton properties 1864
XMLComponent 1866
XMLMethodName 1862
XMLUseParent 1866
TDataSetProvider class 3145
about TDataSetProvider class 3145
ActiveUpdateException 3159
AfterApplyUpdates 3164
AfterExecute 3165
AfterGetParams 3166
AfterGetRecords 3166
AfterRowRequest 3167
AfterUpdateRecord 3168
BeforeApplyUpdates 3168
BeforeExecute 3169
BeforeGetParams 3170
BeforeGetRecords 3170
BeforeRowRequest 3172
BeforeUpdateRecord 3172
CheckDataSet 3149
Constraints 3159
Create 3148
CreateDataPacket 3149
CreateResolver 3150
DataSet 3160
Destroy 3148
DoBeforeExecute 3150
DoBeforeGetRecords 3151
DoGetProviderAttributes 3152
DoGetTableName 3152
Exported 3160
FetchDetails 3153
FindRecord 3153
GetDataSetFromDelta 3153
InternalApplyUpdates 3154
InternalExecute 3154
InternalGetParams 3155
InternalGetRecords 3155
InternalRowRequest 3156
LocateRecord 3157
Notification 3157
OnDataRequest 3173
OnGetData 3174
OnGetDataSetProperties 3174
OnGetTableName 3175
OnUpdateData 3175
OnUpdateError 3176
OnValidate 3176
Options 3161
OwnerData 3162
Params 3162

Reset 3157
ResolveToDataSet 3162
SetCommandText 3158
SetParams 3158
TDataSetProvider events 3163
TDataSetProvider Members 3146
TDataSetProvider methods 3148
TDataSetProvider properties 3159
TransactionStarted 3163
UpdateMode 3163
UpdateRecord 3158
TDataSetResolver class 3176
about TDataSetResolver class 3176
BeginUpdate 3178
Create 3178
DoDelete 3179
DoInsert 3179
DoUpdate 3180
EndUpdate 3180
InitializeConflictBuffer 3181
InternalBeforeResolve 3181
Provider 3182
TDataSetResolver Members 3177
TDataSetResolver methods 3178
TDataSetResolver properties 3181
TDCOMConnection class 1284
about TDCOMConnection class 1284
ComputerName 1287
Create 1286
DoConnect 1287
ObjectBroker 1288
TDCOMConnection Members 1285
TDCOMConnection methods 1286
TDCOMConnection properties 1287
TDeleteButton class 1867
about TDeleteButton class 1867
Create 1869
TDeleteButton Members 1867
TDispatchAppServer class 1289
about TDispatchAppServer class 1289
Create 1289
SafeCallException 1290
TDispatchAppServer Members 1289
TDispatchAppServer methods 1289
TDispatchConnection class 1290
about TDispatchConnection class 1290
AfterConnect 1302
AfterDisconnect 1302
BeforeConnect 1303
BeforeDisconnect 1303
Connected 1296
Create 1291
DoDisconnect 1292
GetConnected 1293
GetServer 1293
GetServerCLSID 1293
GetServerList 1294
LoginPrompt 1296
Notification 1294
ObjectBroker 1300
OnGetUsername 1304
OnLogin 1304
ServerGUID 1301
ServerName 1301
SetConnected 1295
TDispatchConnection events 1302
TDispatchConnection Members 1291
TDispatchConnection methods 1292
TDispatchConnection properties 1295
TempDir
IOPConvert interface 2739
TOPToSoapDomConvert class 2757
TEnumPropDesc class 2631
about TEnumPropDesc class 2631
Create 2632
Destroy 2633
GetStrings 2633
StringToValue 2634

TEnumPropDesc Members 2632
TEnumPropDesc methods 2633
ValueToString 2634
TerminateExtension
TISAPIApplication class 963
TestConnected
TIBDataBase class 188
TEventDispatch class 2634
_AddRef 2636
_Release 2636
about TEventDispatch class 2634
Control 2638
Create 2635
GetIDsOfNames 2636
GetTypeInfo 2636
GetTypeInfoCount 2637
Invoke 2637
QueryInterface 2638
TEventDispatch Members 2635
TEventDispatch methods 2635
TEventDispatch properties 2638
Text
IQueryField interface 2219
TCustomMaskEdit class 1097
TIBQuery class 392
TMaskEdit class 1136
TOutlineNode class 2955
TQueryRadioGroup class 1987
TQuerySelectOptions class 1999
TQueryText class 2010
TRibbonComboBox class 3645
TextHint
TMaskEdit class 1139
TRibbonComboBox class 3646
tfBlob enumeration member 15
tfBytes enumeration member 2382
tfConnect enumeration member 15
tfError enumeration member 15
tfFrames enumeration member 2382
tfHMS enumeration member 2382
TFieldGroup class 1869
about TFieldGroup class 1869
Custom 1873
ImplAddElements 1872
ImplCanAddClass 1872
Style 1873
StyleRule 1874
TFieldGroup Members 1870
TFieldGroup methods 1872
TFieldGroup properties 1873
TFieldRadioGroup class 1874
about TFieldRadioGroup class 1874
Caption 1878
CaptionAttributes 1878
CaptionPosition 1878
Custom 1879
DataSet 1879
DisplayColumns 1880
DisplayWidth 1880
FieldName 1880
Items 1881
ItemsField 1881
ReadOnly 1882
Style 1882
StyleRule 1883
TabIndex 1883
TFieldRadioGroup Members 1875
TFieldRadioGroup properties 1877
Values 1883
ValuesField 1884
TFieldSelectOptions class 1884
about TFieldSelectOptions class 1884
Caption 1888
CaptionAttributes 1888
CaptionPosition 1888
Custom 1889
DataSet 1889
DisplayRows 1890

FieldName 1890
Items 1891
ItemsField 1891
Style 1891
StyleRule 1892
TabIndex 1892
TFieldSelectOptions Members 1885
TFieldSelectOptions properties 1887
Values 1893
ValuesField 1893
TFieldStatus class 1894
about TFieldStatus class 1894
Caption 1896
CaptionAttributes 1896
CaptionPosition 1897
Custom 1897
Style 1897
StyleRule 1898
TFieldStatus Members 1894
TFieldStatus properties 1896
TFieldText class 1899
about TFieldText class 1899
Caption 1901
CaptionAttributes 1902
CaptionPosition 1902
Custom 1902
DisplayWidth 1903
FieldName 1903
ReadOnly 1904
Style 1904
StyleRule 1904
TabIndex 1905
TFieldText Members 1899
TFieldText properties 1901
TFieldTextArea class 1905
about TFieldTextArea class 1905
Caption 1908
CaptionAttributes 1908
CaptionPosition 1909
Custom 1909
DisplayRows 1910
DisplayWidth 1910
FieldName 1910
ReadOnly 1911
Style 1911
StyleRule 1911
TabIndex 1912
TFieldTextArea Members 1906
TFieldTextArea properties 1908
Wrap 1912
TFileStream class 2993
about TFileStream class 2993
Create 2994
Destroy 2994
Read 2995
Seek 2995
SetSize 2995
TFileStream Members 2994
TFileStream methods 2995
Write 2996
TFirstButton class 1913
about TFirstButton class 1913
Create 1915
TFirstButton Members 1913
tfMilliseconds enumeration member 2382
tfMisc enumeration member 15
tfMSF enumeration member 2382
TFormLayout class 1915
about TFormLayout class 1915
BreakButtons 1922
ColumnCount 1922
Create 1916
EndButtons 1917
EndFields 1917
EndLayout 1917
EndRow 1918
EndTable 1918
ImplLayoutButton 1918

ImplLayoutField 1919
ImplLayoutLabelAndField 1919
ImplLayoutTable 1920
NextColumn 1920
StartButtons 1920
StartFields 1921
StartRow 1921
StartTable 1921
TableHeader 1922
TFormLayout Members 1916
TFormLayout methods 1917
TFormLayout properties 1921
tfQExecute enumeration member 15
tfQFetch enumeration member 15
tfQPrepare enumeration member 15
tfSamples enumeration member 2382
tfService enumeration member 15
tfSMPTE24 enumeration member 2382
tfSMPTE25 enumeration member 2382
tfSMPTE30 enumeration member 2382
tfSMPTE30Drop enumeration member 2382
tfStmt enumeration member 15
tfTMSF enumeration member 2382
tfTransact enumeration member 15
TGridAttributes class 1923
about TGridAttributes class 1923
Align 1926
AssignTo 1924
BgColor 1926
Border 1926
CellPadding 1927
CellSpacing 1927
Create 1924
Custom 1928
Parent 1928
Style 1928
StyleRule 1929
TGridAttributes Members 1923
TGridAttributes methods 1924
TGridAttributes properties 1925
TGridRowAttributes class 1929
about TGridRowAttributes class 1929
Align 1932
AssignTo 1931
BgColor 1932
Create 1930
Custom 1933
Parent 1933
Style 1934
StyleRule 1934
TGridRowAttributes Members 1930
TGridRowAttributes methods 1931
TGridRowAttributes properties 1932
VAlign 1934
THashedStringList class 749
about THashedStringList class 749
Changed 751
Destroy 750
IndexOf 751
IndexOfName 752
THashedStringList Members 750
THashedStringList methods 751
THeaderList class 823
about THeaderList class 823
Add 825
Add method 825
Clear 826
Count 828
Create 824
Destroy 824
Extract 826
GetCount 826
GetHeader 827
GetOwnsObjects 827
Headers 828
IndexOf 827
OwnsObjects 829
SetOwnsObjects 827

THeaderList Members 823
THeaderList methods 825
THeaderList properties 827
THelpButton class 3459
about THelpButton class 3459
DrawImage 3460
THelpButton Members 3460
THelpButton methods 3460
ThreadEnded
TIBEvents class 262
TIBBackupRestoreService class 406
about TIBBackupRestoreService class 406
TIBBackupRestoreService Members 406
TIBBackupRestoreService properties 407
Verbose 408
TIBBackupService class 408
about TIBBackupService class 408
BackupFile 411
BlockingFactor 411
Create 410
DatabaseName 411
Destroy 410
Options 411
SetServiceStartOptions 410
TIBBackupService Members 408
TIBBackupService methods 410
TIBBackupService properties 410
TIBBase class 165
about TIBBase class 165
AfterDatabaseDisconnect 175
AfterTransactionEnd 175
BeforeDatabaseDisconnect 175
BeforeTransactionEnd 175
CheckDatabase 170
CheckTransaction 170
Create 166
Database 173
DBHandle 173
Destroy 167
DoAfterDatabaseDisconnect 171
DoAfterTransactionEnd 171
DoBeforeDatabaseDisconnect 171
DoBeforeTransactionEnd 171
DoDatabaseFree 171
DoTransactionFree 172
FAfterDatabaseDisconnect 167
FAfterTransactionEnd 167
FBeforeDatabaseDisconnect 168
FBeforeTransactionEnd 168
FDatabase 168
FIndexInDatabase 168
FIndexInTransaction 169
FOnDatabaseFree 169
FOnTransactionFree 169
FOwner 169
FTransaction 169
GetDBHandle 172
GetTRHandle 172
OnDatabaseFree 176
OnTransactionFree 176
Owner 174
SetDatabase 172
SetTransaction 173
TIBBase events 174
TIBBase fields 167
TIBBase Members 165
TIBBase methods 170
TIBBase properties 173
Transaction 174
TRHandle 174
TIBBatch class 494
about TIBBatch class 494
Columns 496
FColumns 495
FFilename 495
Filename 497
FParams 496
Params 497

ReadyFile 496
TIBBatch fields 495
TIBBatch Members 494
TIBBatch methods 496
TIBBatch properties 496
TIBBatchInput class 497
about TIBBatchInput class 497
ReadParameters 498
TIBBatchInput Members 498
TIBBatchInput methods 498
TIBBatchMove class 26
AbortOnKeyViol 30
AbortOnProblem 30
about TIBBatchMove class 26
ChangedCount 31
ChangedTableName 31
CommitCount 31
Create 27
Destination 32
Destroy 28
Execute 28
KeyViolCount 32
KeyViolTableName 33
Mappings 33
Mode 34
MovedCount 34
Notification 29
ProblemCount 35
ProblemTableName 35
RecordCount 36
Source 36
TIBBatchMove Members 26
TIBBatchMove methods 28
TIBBatchMove properties 29
Transliterate 37
TIBBatchOutput class 498
about TIBBatchOutput class 498
TIBBatchOutput Members 499
TIBBatchOutput methods 499
WriteColumns 499
TIBBCDField class 59
about TIBBCDField class 59
CheckTypeSize 60
Create 60
GetAsCurrency 61
GetAsString 61
GetAsVariant 61
GetDataSize 61
Size 62
TIBBCDField Members 59
TIBBCDField methods 60
TIBBCDField properties 61
TIBBlobStream class 40
about TIBBlobStream class 40
BlobID 50
BlobMaxSegmentSize 51
BlobNumSegments 51
BlobSize 51
BlobType 51
Call 43
Cancel 43
CheckReadable 43
CheckWritable 44
CloseBlob 44
Create 41
CreateBlob 44
Database 52
DBHandle 52
Destroy 42
EnsureBlobInitialized 44
Finalize 45
GetBlobInfo 45
GetDatabase 45
GetDBHandle 45
GetTransaction 45
GetTRHandle 46
Handle 52
LoadFromFile 46

LoadFromStream 46
Mode 52
Modified 53
OpenBlob 46
Read 47
SaveToFile 47
SaveToStream 47
Seek 48
SetBlobID 48
SetDatabase 48
SetMode 49
SetSize 49
SetTransaction 49
TIBBlobStream Members 40
TIBBlobStream methods 42
TIBBlobStream properties 50
Transaction 53
TRHandle 53
Truncate 49
Write 50
TIBConfigService class 412
about TIBConfigService class 412
ActivateShadow 414
BringDatabaseOnline 414
DatabaseName 417
ServiceStart 414
SetAsyncMode 415
SetDBSqlDialect 415
SetPageBuffers 415
SetReadOnly 416
SetReserveSpace 416
SetSweepInterval 416
ShutdownDatabase 417
TIBConfigService Members 412
TIBConfigService methods 413
TIBConfigService properties 417
TIBControlAndQueryService class 417
about TIBControlAndQueryService class 417
Action 420
BufferSize 420
Create 419
Eof 421
GetNextChunk 419
GetNextLine 420
TIBControlAndQueryService Members 418
TIBControlAndQueryService methods 419
TIBControlAndQueryService properties 420
TIBControlService class 421
about TIBControlService class 421
Create 422
InternalServiceStart 423
IsServiceRunning 424
ServiceStart 423
ServiceStartAddParam 424
ServiceStartAddParam method 423
ServiceStartParams 425
SetServiceStartOptions 424
TIBControlService Members 421
TIBControlService methods 423
TIBControlService properties 424
TIBCustomDataSet class 62
about TIBCustomDataSet class 62
ActivateConnection 69
ActivateTransaction 69
AfterCancel 94
AfterClose 94
AfterDatabaseDisconnect 109
AfterDelete 94
AfterEdit 94
AfterInsert 95
AfterOpen 95
AfterPost 95
AfterRefresh 95
AfterScroll 95
AfterTransactionEnd 109
AllocRecordBuffer 69
ApplyUpdates 69
AutoCalcFields 96

BeforeCancel 96
BeforeClose 96
BeforeDatabaseDisconnect 109
BeforeDelete 96
BeforeEdit 97
BeforeInsert 97
BeforeOpen 97
BeforePost 97
BeforeRefresh 97
BeforeScroll 98
BeforeTransactionEnd 109
BookmarkValid 70
BufferChunks 98
CachedUpdates 98
CachedUpdateStatus 70
CancelUpdates 70
CheckDatasetClosed 71
CheckDatasetOpen 71
CheckNotUniDirectional 71
ClearCalcFields 71
CompareBookmarks 71
ConstraintsStored 72
Create 66
CreateBlobStream 72
CreateFields 72
Current 73
Database 99
DatabaseFree 110
DBHandle 99
DeactivateTransaction 73
DeleteSQL 99
Destroy 66
Disconnect 73
DoBeforeDelete 73
DoBeforeEdit 73
DoBeforeInsert 74
DoOnNewRecord 74
FetchAll 74
ForcedRefresh 99
FreeRecordBuffer 75
GeneratorField 100
GetActiveBuf 75
GetBookmarkData 75
GetBookmarkFlag 75
GetCanModify 75
GetCurrentRecord 76
GetDataSource 76
GetFieldClass 76
GetFieldData 77
GetFieldData method 76
GetRecNo 77
GetRecord 77
GetRecordCount 78
GetRecordSize 78
InitRecord 78
InsertSQL 100
InternalAddRecord 78
InternalBatchInput 79
InternalBatchOutput 79
InternalCancel 79
InternalClose 79
InternalDelete 79
InternalExecQuery 80
InternalFirst 80
InternalGotoBookmark 80
InternalHandleException 80
InternalInitFieldDefs 81
InternalInitRecord 81
InternalInsert 81
InternalLast 81
InternalOpen 81
InternalPost 82
InternalPrepare 82
InternalPrepared 100
InternalRefresh 82
InternalRefreshRow 82
InternalSetParamsFromCursor 83
InternalSetToRecord 83

InternalUnPrepare 83
IsCursorOpen 83
IsSequenced 83
LiveMode 101
Loaded 84
Locate 84
LocateNext 85
Lookup 85
ModifySQL 101
ObjectView 101
OnCalcFields 101
OnDeleteError 102
OnEditError 102
OnNewRecord 102
OnPostError 102
OnUpdateError 110
OnUpdateRecord 111
ParamCheck 102
Params 103
Plan 103
Post 86
PSEndTransaction 86
PSExecuteStatement 86
PSGetQuoteChar 87
PSGetTableName 87
PSGetUpdateException 87
PSInTransaction 87
PSIsSQLBased 88
PSIsSQLSupported 88
PSReset 88
PSStartTransaction 88
PSUpdateRecord 88
QDelete 103
QInsert 104
QModify 104
QRefresh 104
QSelect 104
RecordModified 89
RefreshSQL 105
ReQuery 89
RevertRecord 89
RowsAffected 105
SelectSQL 105
SelectStmtHandle 105
SetActive 89
SetBookmarkData 90
SetBookmarkFlag 90
SetCachedUpdates 90
SetDataSource 90
SetFieldData 91
SetFieldData method 90
SetRecNo 91
SQLParams 106
SQLType 91
StatementType 106
TIBCustomDataSet events 108
TIBCustomDataSet Members 62
TIBCustomDataSet methods 67
TIBCustomDataSet properties 92
Transaction 106
TransactionFree 111
TRHandle 106
Undelete 92
UniDirectional 107
UpdateMode 107
UpdateObject 107
UpdateRecordTypes 108
UpdatesPending 108
UpdateStatus 92
TIBCustomService class 425
about TIBCustomService class 425
Active 430
Attach 427
BufferSize 430
CheckActive 427
CheckInactive 428
Create 426
Destroy 426

Detach 428
Handle 430
InternalServiceQuery 428
Loaded 428
Login 429
LoginPrompt 430
OnAttach 433
OnLogin 433
OutputBuffer 431
OutputBufferOption 431
Params 431
Protocol 431
ServerName 431
ServiceParamBySPB 432
ServiceQueryParams 432
TIBCustomService events 432
TIBCustomService Members 425
TIBCustomService methods 427
TIBCustomService properties 429
TraceFlags 432
TIBCustomSQLMonitor class 561
about TIBCustomSQLMonitor class 561
Create 562
Destroy 562
Enabled 563
Handle 563
OnSQL 564
Release 562
TIBCustomSQLMonitor events 564
TIBCustomSQLMonitor Members 561
TIBCustomSQLMonitor methods 562
TIBCustomSQLMonitor properties 563
TraceFlags 564
TIBDataBase class 176
about TIBDataBase class 176
AddEventNotifier 180
AddTransaction 180
AfterConnect 198
AfterDisconnect 199
AllowStreamedConnected 189
ApplyUpdates 180
BeforeConnect 199
BeforeDisconnect 199
Call 181
CheckActive 181
CheckDatabaseName 181
CheckInactive 181
CloseDataSets 182
Connected 189
Create 178
CreateDatabase 182
DatabaseName 190
DBParamByDPB 190
DBSQLDialect 190
DefaultTransaction 191
Destroy 178
DoConnect 182
DoDisconnect 183
DropDatabase 183
FindDefaultTransaction 183
FindTransaction 184
FlushSchema 184
ForceClose 184
GetConnected 184
GetFieldNames 185
GetTableNames 185
Handle 191
HandleIsShared 191
Has_COMPUTED_BLR 185
Has_DEFAULT_VALUE 186
IdleTimer 191
IndexOfDBConst 186
InternalTransaction 192
IsReadOnly 192
Loaded 186
LoginPrompt 192
Notification 187
OnDialectDowngradeWarning 200

OnIdleTimer 200
OnLogin 201
Params 196
RemoveEventNotifier 187
RemoveTransaction 187
RemoveTransactions 188
SetHandle 188
SQLDialect 196
SQLObjectCount 197
SQLObjects 197
TestConnected 188
TIBDataBase events 198
TIBDataBase Members 177
TIBDataBase methods 179
TIBDataBase properties 188
TraceFlags 197
TransactionCount 198
Transactions 198
TIBDatabaseInfo class 232
about TIBDatabaseInfo class 232
Allocation 247
BackoutCount 247
BaseLevel 248
Call 238
Create 234
CurrentMemory 248
Database 248
DBFileName 248
DBImplementationClass 249
DBImplementationNo 249
DBSiteName 249
DBSQLDialect 249
DeleteCount 250
Destroy 234
ExpungeCount 250
FBackoutCount 235
FDatabase 235
FDeleteCount 236
Fetches 250
FExpungeCount 236
FIBLoaded 236
FInsertCount 236
ForcedWrites 250
FPurgeCount 236
FReadIdxCount 237
FReadSeqCount 237
FUpdateCount 237
FUserNames 237
GetAllocation 239
GetBackoutCount 239
GetBaseLevel 239
GetCurrentMemory 239
GetDBFileName 239
GetDBImplementationClass 240
GetDBImplementationNo 240
GetDBSiteName 240
GetDBSQLDialect 240
GetDeleteCount 241
GetExpungeCount 241
GetFetches 241
GetForcedWrites 241
GetInsertCount 241
GetLongDatabaseInfo 242
GetMarks 242
GetMaxMemory 242
GetNoReserve 242
GetNumBuffers 243
GetODSMajorVersion 243
GetODSMinorVersion 243
GetOperationCounts 243
GetPageSize 244
GetPurgeCount 244
GetReadIdxCount 244
GetReadOnly 244
GetReads 244
GetReadSeqCount 245
GetStringDatabaseInfo 245
GetSweepInterval 245

GetUpdateCount 245
GetUserNames 246
GetVersion 246
GetWrites 246
InsertCount 251
Marks 251
MaxMemory 251
NoReserve 251
NumBuffers 252
ODSMajorVersion 252
ODSMinorVersion 252
PageSize 252
PurgeCount 253
ReadIdxCount 253
ReadOnly 253
Reads 253
ReadSeqCount 254
SweepInterval 254
TIBDatabaseInfo fields 235
TIBDatabaseInfo Members 232
TIBDatabaseInfo methods 237
TIBDatabaseInfo properties 246
UpdateCount 254
UserNames 255
Version 255
Writes 255
TIBDataLink class 111
about TIBDataLink class 111
ActiveChanged 113
CheckBrowseMode 113
Create 112
Destroy 112
GetDetailDataSet 113
RecordChanged 114
TIBDataLink Members 112
TIBDataLink methods 113
TIBDataSet class 114
about TIBDataSet class 114
Active 122
AfterCancel 122
AfterClose 122
AfterDatabaseDisconnect 133
AfterDelete 123
AfterEdit 123
AfterInsert 123
AfterOpen 123
AfterPost 123
AfterScroll 124
AfterTransactionEnd 133
AutoCalcFields 124
BatchInput 119
BatchOutput 119
BeforeCancel 124
BeforeClose 124
BeforeDatabaseDisconnect 133
BeforeDelete 125
BeforeEdit 125
BeforeInsert 125
BeforeOpen 125
BeforePost 125
BeforeScroll 126
BeforeTransactionEnd 134
BufferChunks 126
CachedUpdates 126
DatabaseFree 134
DataSource 127
DeleteSQL 127
ExecSQL 119
Filtered 127
GeneratorField 127
InsertSQL 128
InternalOpen 119
LiveMode 128
ModifySQL 128
OnCalcFields 129
OnDeleteError 129
OnEditError 129
OnFilterRecord 129

OnNewRecord 129
OnPostError 130
ParamByName 120
ParamCheck 130
Params 130
Prepare 120
Prepared 131
PSSetCommandText 120
RefreshSQL 131
SelectSQL 131
SelectStmtHandle 131
SetFiltered 120
StatementType 132
TIBDataSet events 133
TIBDataSet Members 114
TIBDataSet methods 118
TIBDataSet properties 121
TransactionFree 134
UniDirectional 132
UnPrepare 121
UpdateObject 132
TIBDataSetUpdateObject class 134
about TIBDataSetUpdateObject class 134
Apply 136
Create 135
DataSet 137
Destroy 136
GetDataSet 136
GetSQL 137
RefreshSQL 137
SetDataSet 137
TIBDataSetUpdateObject Members 135
TIBDataSetUpdateObject methods 136
TIBDataSetUpdateObject properties 137
TIBDSBlobStream class 138
about TIBDSBlobStream class 138
Create 139
Destroy 139
FBlobStream 139
FField 140
FModified 140
Read 140
Seek 141
SetSize 141
TIBDSBlobStream fields 139
TIBDSBlobStream Members 138
TIBDSBlobStream methods 140
Write 141
TIBEvents class 258
about TIBEvents class 258
AutoRegister 263
Create 259
Database 263
Destroy 260
EventChange 260
Events 264
GetAutoRegister 260
GetNativeHandle 261
NativeHandle 264
Notification 261
OnError 265
OnEventAlert 265
Registered 264
RegisterEvents 261
SetAutoRegister 262
ThreadEnded 262
TIBEvents events 265
TIBEvents Members 258
TIBEvents methods 260
TIBEvents properties 263
UnRegisterEvents 262
ValidateDatabase 263
TIBExtract class 271
about TIBExtract class 271
Create 272
Database 281
DatabaseInfo 281
Destroy 273

ExtractDDL 274
ExtractListTable 274
ExtractListView 274
ExtractObject 274
GetArrayField 276
GetCharacterSets 276
GetFieldType 276
Items 281
ListAllTables 276
ListCheck 277
ListCreateDb 277
ListData 277
ListDomains 277
ListException 278
ListFilters 278
ListForeign 278
ListFunctions 278
ListGenerators 278
ListGrants 279
ListIndex 279
ListProcs 279
ListRoles 279
ListTriggers 280
ListViews 280
Notification 280
PrintSet 280
ShowSystem 282
TIBExtract Members 271
TIBExtract methods 273
TIBExtract properties 281
Transaction 282
TIBGeneratorField class 142
about TIBGeneratorField class 142
Apply 143
ApplyEvent 145
Assign 144
Create 143
Field 145
Generator 145
IncrementBy 146
TIBGeneratorField Members 142
TIBGeneratorField methods 143
TIBGeneratorField properties 144
ValueName 144
TIBInputDelimitedFile class 500
about TIBInputDelimitedFile class 500
ColDelimiter 504
Destroy 501
FColDelimiter 501
FEOF 501
FFile 502
FLookAhead 502
FReadBlanksAsNull 502
FRowDelimiter 502
FSkipTitles 503
GetColumn 503
ReadBlanksAsNull 504
ReadParameters 503
ReadyFile 503
RowDelimiter 504
SkipTitles 505
TIBInputDelimitedFile fields 501
TIBInputDelimitedFile Members 500
TIBInputDelimitedFile methods 503
TIBInputDelimitedFile properties 504
TIBInputRawFile class 505
about TIBInputRawFile class 505
Destroy 506
FFile 506
ReadParameters 507
ReadyFile 507
TIBInputRawFile fields 506
TIBInputRawFile Members 505
TIBInputRawFile methods 506
TIBInstall class 310
about TIBInstall class 310
Create 311
DestinationDirectory 315

Destroy 312
GetOptionDescription 312
GetOptionDescription method 312
GetOptionName 313
GetOptionName method 313
GetOptionSpaceRequired 314
GetOptionSpaceRequired method 313
InstallCheck 314
InstallExecute 314
InstallOptions 315
SourceDirectory 316
SuggestedDestination 316
TIBInstall Members 310
TIBInstall methods 312
TIBInstall properties 315
UnInstallFile 316
TIBLicensingService class 433
about TIBLicensingService class 433
Action 436
AddLicense 435
ID 436
Key 437
RemoveLicense 435
SetServiceStartOptions 436
TIBLicensingService Members 434
TIBLicensingService methods 435
TIBLicensingService properties 436
TIBLogService class 437
about TIBLogService class 437
SetServiceStartOptions 439
TIBLogService Members 437
TIBLogService methods 438
TIBOutputDelimitedFile class 507
about TIBOutputDelimitedFile class 507
ColDelimiter 510
Destroy 508
FColDelimiter 509
FFile 509
FOutputTitles 509
FRowDelimiter 509
OutputTitles 511
ReadyFile 510
RowDelimiter 511
TIBOutputDelimitedFile fields 508
TIBOutputDelimitedFile Members 508
TIBOutputDelimitedFile methods 509
TIBOutputDelimitedFile properties 510
WriteColumns 510
TIBOutputRawFile class 511
about TIBOutputRawFile class 511
Destroy 512
FFile 512
ReadyFile 513
TIBOutputRawFile fields 512
TIBOutputRawFile Members 512
TIBOutputRawFile methods 513
WriteColumns 513
TIBQuery class 376
about TIBQuery class 376
Active 387
AfterDatabaseDisconnect 394
AfterTransactionEnd 394
BatchInput 382
BatchOutput 382
BeforeDatabaseDisconnect 394
BeforeTransactionEnd 395
BufferChunks 387
CachedUpdates 388
Constraints 388
Create 381
DatabaseFree 395
DataSource 388
DefineProperties 382
Destroy 381
Disconnect 383
ExecSQL 383
Filtered 389
GenerateParamNames 389

GenerateQueryForLiveUpdate 383
GeneratorField 389
GetDetailLinkFields 383
GetParamsCount 384
InitFieldDefs 384
InternalOpen 384
LiveMode 389
OnFilterRecord 390
ParamByName 384
ParamCheck 390
ParamCount 390
Params 391
Prepare 385
Prepared 391
PSExecute 385
PSGetParams 385
PSGetTableName 385
PSSetCommandText 386
PSSetParams 386
RowsAffected 391
SetFiltered 386
SQL 392
StatementType 392
StmtHandle 392
Text 392
TIBQuery events 394
TIBQuery Members 376
TIBQuery methods 381
TIBQuery properties 387
TransactionFree 395
UniDirectional 393
UnPrepare 386
UpdateObject 393
TIBRestoreService class 439
about TIBRestoreService class 439
BackupFile 442
Create 440
DatabaseName 442
Destroy 441
Options 442
PageBuffers 442
PageSize 443
SetServiceStartOptions 441
TIBRestoreService Members 439
TIBRestoreService methods 441
TIBRestoreService properties 441
TIBSchema class 201
about TIBSchema class 201
FreeNodes 202
Has_COMPUTED_BLR 202
Has_DEFAULT_VALUE 202
TIBSchema Members 201
TIBSchema methods 202
TIBSecurityService class 443
about TIBSecurityService class 443
AddUser 446
Create 445
DeleteUser 446
Destroy 445
DisplayUser 446
DisplayUsers 447
FirstName 449
GroupID 449
LastName 449
Loaded 447
MiddleName 449
ModifyUser 448
Password 450
SecurityAction 450
SetServiceStartOptions 448
SQlRole 450
TIBSecurityService Members 444
TIBSecurityService methods 446
TIBSecurityService properties 448
UserID 450
UserInfo 451
UserInfoCount 451
UserName 451

TIBServerProperties class 452
about TIBServerProperties class 452
ConfigParams 456
Create 453
DatabaseInfo 456
Destroy 453
Fetch 454
FetchConfigParams 454
FetchDatabaseInfo 455
FetchLicenseInfo 455
FetchLicenseMaskInfo 455
FetchVersionInfo 455
LicenseInfo 457
LicenseMaskInfo 457
Options 457
TIBServerProperties Members 452
TIBServerProperties methods 454
TIBServerProperties properties 456
VersionInfo 457
TIBSetup class 317
about TIBSetup class 317
Call 318
Create 318
ErrorContext 319
ErrorInternal 318
GetInstallMessage 319
IBInstallError 319
MsgFilePath 320
OnError 321
OnStatusChange 322
OnWarning 322
Progress 320
RebootToComplete 320
StatusContext 321
StatusInternal 319
TIBSetup events 321
TIBSetup Members 317
TIBSetup methods 318
TIBSetup properties 319
TIBSQL class 513
about TIBSQL class 513
BatchInput 521
BatchOutput 522
BeforeTransactionEnd 522
Bof 530
Call 522
CheckClosed 522
CheckOpen 523
CheckValidStatement 523
Close 523
Create 516
Current 523
Database 530
DBHandle 530
Destroy 516
DoBeforeDatabaseDisconnect 524
Eof 530
ExecQuery 524
FBase 517
FBOF 517
FCursor 517
FEOF 517
FGenerateParamNames 518
FGoToFirstRecordOnExecute 518
FHandle 518
FieldByName 524
FieldIndex 531
Fields 531
FOnSQLChanging 518
FOpen 519
FParamCheck 519
FPrepared 519
FProcessedSQL 519
FRecordCount 519
FreeHandle 524
FSQL 520
FSQLParams 520
FSQLRecord 520

FSQLType 520
GenerateParamNames 531
GetDatabase 525
GetDBHandle 525
GetEOF 525
GetFieldIndex 525
GetFields 525
GetPlan 526
GetRecordCount 526
GetRowsAffected 526
GetSQLParams 526
GetTransaction 527
GetTRHandle 527
GetUniqueRelationName 527
GoToFirstRecordOnExecute 532
Handle 532
Next 527
OnSQLChanging 535
Open 532
ParamByName 527
ParamCheck 532
Params 533
Plan 533
Prepare 528
Prepared 533
PreprocessSQL 528
RecordCount 533
RowsAffected 534
SetDatabase 528
SetSQL 528
SetTransaction 529
SQL 534
SQLChanging 529
SQLType 534
TIBSQL events 535
TIBSQL fields 516
TIBSQL Members 514
TIBSQL methods 520
TIBSQL properties 529
Transaction 535
TRHandle 535
UniqueRelationName 535
TIBSQLMonitor class 564
about TIBSQLMonitor class 564
Enabled 565
OnSQL 566
TIBSQLMonitor events 566
TIBSQLMonitor Members 565
TIBSQLMonitor properties 565
TraceFlags 566
TIBStatisticalService class 458
about TIBStatisticalService class 458
DatabaseName 460
Options 460
SetServiceStartOptions 460
TIBStatisticalService Members 458
TIBStatisticalService methods 459
TIBStatisticalService properties 460
TIBStoredProc class 582
about TIBStoredProc class 582
AfterDatabaseDisconnect 595
AfterTransactionEnd 595
BeforeDatabaseDisconnect 596
BeforeTransactionEnd 596
CopyParams 588
Create 586
DatabaseFree 596
DefineProperties 588
Destroy 587
Disconnect 588
ExecProc 588
Filtered 592
GetParamsCount 589
InternalOpen 589
OnFilterRecord 593
ParamByName 589
ParamCount 593
Params 593

Prepare 590
Prepared 594
PSExecute 590
PSGetParams 590
PSGetTableName 590
PSSetCommandText 591
PSSetParams 591
SetFiltered 591
SetPrepare 591
SetPrepared 591
SetProcName 592
StmtHandle 594
StoredProcedureNames 594
StoredProcName 595
TIBStoredProc events 595
TIBStoredProc Members 582
TIBStoredProc methods 587
TIBStoredProc properties 592
TransactionFree 596
UnPrepare 592
TIBStringField class 146
about TIBStringField class 146
CheckTypeSize 147
Create 147
GetAsString 148
GetAsVariant 148
GetValue 148
SetAsString 148
TIBStringField Members 146
TIBStringField methods 147
TIBTable class 600
about TIBTable class 600
Active 615
AddIndex 606
AfterDatabaseDisconnect 623
AfterTransactionEnd 623
BeforeDatabaseDisconnect 624
BeforeTransactionEnd 624
BufferChunks 615
CachedUpdates 615
Constraints 616
Create 605
CreateTable 606
CurrentDBKey 616
DatabaseFree 624
DataEvent 607
DefaultIndex 616
DefChanged 607
DeleteIndex 607
DeleteTable 608
Destroy 605
DoOnNewRecord 608
EmptyTable 608
Exists 616
FieldDefs 617
Filter 617
Filtered 617
GetCanModify 608
GetDataSource 609
GetDetailLinkFields 609
GetIndexNames 609
GetIndexParams 609
GotoCurrent 610
IndexDefs 618
IndexFieldCount 618
IndexFieldNames 618
IndexFields 619
IndexName 619
InitFieldDefs 610
InternalClose 610
InternalOpen 610
InternalRefresh 611
InternalRefreshRow 611
LiveMode 619
Locate 611
MasterFields 620
MasterSource 620
OnFilterRecord 620

PSGetDefaultOrder 612
PSGetIndexDefs 612
PSGetKeyFields 612
PSGetTableName 612
PSSetCommandText 613
PSSetParams 613
ReadOnly 621
SetFiltered 613
SetFilterOptions 613
SetFilterText 614
StoreDefs 621
TableName 621
TableNames 622
TableTypes 622
TIBTable events 623
TIBTable Members 600
TIBTable methods 605
TIBTable properties 614
TransactionFree 624
UniDirectional 622
UpdateIndexDefs 614
UpdateObject 623
TIBTimer class 202
about TIBTimer class 202
Create 203
Destroy 204
Enabled 205
Interval 206
OnTimer 206
TIBTimer events 206
TIBTimer Members 203
TIBTimer methods 205
TIBTimer properties 205
Timer 205
TIBTransaction class 206
about TIBTransaction class 206
Active 215
AddDatabase 209
AutoStopAction 215
Call 209
CheckAutoStop 210
CheckDatabasesInList 210
CheckInTransaction 210
CheckNotInTransaction 210
Commit 211
CommitRetaining 211
Create 208
DatabaseCount 216
Databases 216
DefaultAction 216
DefaultDatabase 217
Destroy 208
FindDatabase 211
FindDefaultDatabase 212
Handle 217
HandleIsShared 217
IdleTimer 217
InTransaction 218
Loaded 212
Notification 212
OnIdleTimer 219
Params 218
RemoveDatabase 213
RemoveDatabases 213
Rollback 213
RollbackRetaining 214
SetHandle 214
SQLObjectCount 218
SQLObjects 218
StartTransaction 214
TIBTransaction events 219
TIBTransaction Members 207
TIBTransaction methods 208
TIBTransaction properties 215
TPB 219
TPBLength 219
TIBUnInstall class 322
about TIBUnInstall class 322

TIBUnInstall Members 323
TIBUnInstall methods 323
TIBUnInstall properties 324
UnInstallCheck 324
UnInstallExecute 324
UnInstallFile 324
TIBUpdateSQL class 630
about TIBUpdateSQL class 630
Apply 632
Create 631
DataSet 635
DeleteSQL 635
Destroy 631
ExecSQL 633
GetDataSet 633
GetSQL 633
InsertSQL 636
ModifySQL 636
Query 636
SetDataSet 634
SetParams 634
SQL 637
SQLChanged 634
TIBUpdateSQL Members 630
TIBUpdateSQL methods 632
TIBUpdateSQL properties 634
TIBValidationService class 461
about TIBValidationService class 461
Create 462
DatabaseName 464
Destroy 463
FetchLimboTransactionInfo 463
FixLimboTransactionErrors 463
GlobalAction 465
LimboTransactionInfo 465
LimboTransactionInfoCount 465
Options 465
SetServiceStartOptions 464
TIBValidationService Members 461
TIBValidationService methods 463
TIBValidationService properties 464
TIBXSQLDA class 536
about TIBXSQLDA class 536
AddName 540
AsXSQLDA 542
ByName 540
Count 543
Create 537
Destroy 537
FCount 538
FNames 538
FSize 538
FSQL 539
FUniqueRelationName 539
FXSQLDA 539
FXSQLVARs 539
GetModified 540
GetNames 541
GetRecordSize 541
GetXSQLDA 541
GetXSQLVAR 541
GetXSQLVARByName 541
Initialize 542
Modified 543
Names 543
RecordSize 543
SetCount 542
TIBXSQLDA fields 538
TIBXSQLDA Members 536
TIBXSQLDA methods 539
TIBXSQLDA properties 542
UniqueRelationName 544
Vars 544
TIBXSQLVAR class 544
about TIBXSQLVAR class 544
AsCurrency 548
AsDate 548
AsDateTime 549

AsDouble 549
AsFloat 549
AsInt64 550
AsInteger 550
AsLong 550
AsPointer 550
AsQuad 551
AsShort 551
Assign 546
AsString 551
AsTime 551
AsTrimString 552
AsVariant 552
AsXSQLVAR 552
Clear 546
Create 545
Data 553
Index 553
IsNull 553
IsNullable 553
LoadFromFile 547
LoadFromStream 547
Modified 554
Name 554
SaveToFile 547
SaveToStream 547
Size 554
SQLType 554
TIBXSQLVAR Members 545
TIBXSQLVAR methods 546
TIBXSQLVAR properties 548
Value 555
TimeBtnState
TTimerSpeedButton class 3911
TimeFormat
TMediaPlayer class 2362
Timer
TIBTimer class 205
TInetXPageElements class 2283
about TInetXPageElements class 2283
BodyContent 2285
FormsContent 2284
IncludesContent 2284
ScriptContent 2284
StylesContent 2284
TInetXPageElements fields 2283
TInetXPageElements Members 2283
TInetXPageElements methods 2285
WarningsContent 2284
TInetXPageProducer class 2285
about TInetXPageProducer class 2285
EnableXMLIslands 2288
HTMLDoc 2288
HTMLFile 2289
IncludePathURL 2290
OnAfterGetContent 2293
OnAfterGetXMLData 2293
OnBeforeGetContent 2294
OnBeforeGetXMLData 2294
ScriptEngine 2290
Styles 2291
StylesFile 2291
TInetXPageProducer events 2292
TInetXPageProducer Members 2286
TInetXPageProducer properties 2287
WebPageItems 2292
TIniFile class 752
about TIniFile class 752
DeleteKey 757
Destroy 756
EraseSection 757
ReadSection 758
ReadSections 758
ReadSectionValues 759
ReadString 759
TIniFile Members 755
TIniFile methods 757
UpdateFile 762

WriteString 762
TInsertButton class 1935
about TInsertButton class 1935
Create 1937
TInsertButton Members 1936
TInstallOptions class 325
about TInstallOptions class 325
CmdLineTools 325
ConnectivityClients 326
Examples 326
MainComponents 326
TInstallOptions Members 325
TInstallOptions properties 325
TInvContext class 829
about TInvContext class 829
AllocServerData 831
GetParamPointer 831
GetResultPointer 832
SetMethodInfo 832
SetParamPointer 833
SetResultPointer 833
TInvContext Members 830
TInvContext methods 831
TInvokableClass class 833
_AddRef 836
_Release 837
about TInvokableClass class 833
AfterConstruction 837
BeforeDestruction 838
Create 835
FRefCount 836
FSOAPHeaders 836
NewInstance 838
QueryInterface 839
RefCount 839
TInvokableClass fields 835
TInvokableClass Members 834
TInvokableClass methods 836
TInvokableClass properties 839
TInvokableClassRegistry class 840
about TInvokableClassRegistry class 840
Create 842
Destroy 842
GetActionURIOfIID 844
GetActionURIOfInfo 844
GetClassFromIntfInfo 845
GetExceptionInfoForInterface 845
GetHeaderClass 846
GetHeaderInfoForInterface 846
GetHeaderName 847
GetHeaderNamespace 848
GetHeaderNamespace method 847
GetInfoForURI 848
GetInterfaceCount 849
GetInterfaceExternalName 849
GetInterfaceExternalName method 849
GetInterfaceInfoFromName 850
GetInterfaceTypeInfo 850
GetIntfInvokeOptions 851
GetIntfInvokeOptions method 851
GetInvokableObjectFromClass 852
GetMethExternalName 852
GetMethInternalName 852
GetNamespaceByGUID 853
GetParamExternalName 853
GetParamInternalName 854
GetRegInterfaceEntry 854
GetRequestHeaderInfoForInterface 855
GetResponseHeaderInfoForInterface 855
GetReturnParamNames 856
GetUDDIInfo 856
GetUDDIInfo method 856
GetWSDLEncoding 857
GetWSDLEncoding method 857
HasRegInterfaceImpl 858
RegisterAllSOAPActions 859
RegisterDefaultSOAPAction 859
RegisterException 860

RegisterExceptionMethod 860
RegisterExternalMethName 861
RegisterExternalParamName 861
RegisterHeaderClass 862
RegisterHeaderClass method 862
RegisterHeaderMethod 863
RegisterInterface 864
RegisterInvokableClass 865
RegisterInvokableClass method 865
RegisterInvokeOptions 866
RegisterInvokeOptions method 866
RegisterReturnParamNames 867
RegisterUDDIInfo 867
TInvokableClassRegistry Members 840
TInvokableClassRegistry methods 842
UnRegisterInterface 867
UnRegisterInvokableClass 868
TISAPIApplication class 960
about TISAPIApplication class 960
Create 961
Destroy 961
GetExtensionVersion 962
HttpExtensionProc 962
TerminateExtension 963
TISAPIApplication Members 961
TISAPIApplication methods 962
TISAPIFactory class 963
about TISAPIFactory class 963
Create 963
NewRequest 964
NewResponse 964
TISAPIFactory Members 963
TISAPIFactory methods 964
TISAPIRequest class 970
about TISAPIRequest class 970
Create 971
ECB 976
GetDateVariable 972
GetFieldByName 972
GetIntegerVariable 973
GetStringVariable 973
ReadClient 974
ReadString 974
TISAPIRequest Members 970
TISAPIRequest methods 971
TISAPIRequest properties 976
TranslateURI 975
WriteClient 975
WriteHeaders 975
WriteString 976
TISAPIResponse class 977
about TISAPIResponse class 977
Create 978
GetContent 979
GetDateVariable 979
GetIntegerVariable 980
GetLogMessage 980
GetStatusCode 980
GetStringVariable 981
InitResponse 981
SendRedirect 982
SendResponse 982
SendStream 982
Sent 983
SetContent 983
SetDateVariable 983
SetIntegerVariable 984
SetLogMessage 984
SetStatusCode 984
SetStringVariable 985
TISAPIResponse Members 977
TISAPIResponse methods 978
Title
TPrinter class 3065
TJPEGData class 992
about TJPEGData class 992
Destroy 993
FreeHandle 993

TJPEGData Members 992
TJPEGData methods 993
TJPEGImage class 994
about TJPEGImage class 994
Assign 998
AssignTo 999
Bitmap 1008
Changed 1000
Compress 1000
CompressionQuality 1008
Create 995
Destroy 997
DIBNeeded 1000
Draw 1001
Equals 1001
FreeBitmap 1001
GetEmpty 1001
GetHeight 1002
GetPalette 1002
GetWidth 1002
Grayscale 1008
JPEGNeeded 1003
LoadFromClipboardFormat 1003
LoadFromStream 1004
NewBitmap 1004
NewImage 1004
Performance 1008
PixelFormat 1009
ProgressiveDisplay 1009
ProgressiveEncoding 1010
ReadData 1004
ReadStream 1005
SaveToClipboardFormat 1005
SaveToStream 1005
Scale 1010
SetHeight 1006
SetPalette 1006
SetWidth 1006
Smoothing 1010
TJPEGImage Members 994
TJPEGImage methods 997
TJPEGImage properties 1007
WriteData 1007
tk_alias enumeration member 2826
tk_any enumeration member 2826
tk_array enumeration member 2826
tk_boolean enumeration member 2826
tk_char enumeration member 2826
tk_double enumeration member 2826
tk_enum enumeration member 2826
tk_except enumeration member 2826
tk_fixed enumeration member 2826
tk_float enumeration member 2826
tk_long enumeration member 2826
tk_longdouble enumeration member 2826
tk_longlong enumeration member 2826
tk_null enumeration member 2826
tk_objref enumeration member 2826
tk_octet enumeration member 2826
tk_Principal enumeration member 2826
tk_sequence enumeration member 2826
tk_short enumeration member 2826
tk_string enumeration member 2826
tk_struct enumeration member 2826
tk_TypeCode enumeration member 2826
tk_ulong enumeration member 2826
tk_ulonglong enumeration member 2826
tk_union enumeration member 2826
tk_ushort enumeration member 2826
tk_void enumeration member 2826
tk_wchar enumeration member 2826
tk_wstring enumeration member 2826
TKeyTipList class 3592
about TKeyTipList class 3592
Add 3594
Clear 3594
Count 3594
Create 3593

Current 3595
Destroy 3593
Full 3595
Items 3595
Partial 3596
TKeyTipList Members 3592
TKeyTipList methods 3593
TKeyTipList properties 3594
TKeyTipListEnumerator class 3596
about TKeyTipListEnumerator class 3596
Create 3597
Current 3598
GetCurrent 3597
MoveNext 3597
TKeyTipListEnumerator Members 3596
TKeyTipListEnumerator methods 3597
TKeyTipListEnumerator properties 3597
TKeyTipMatchItem class 3598
about TKeyTipMatchItem class 3598
Control 3599
Create 3599
Handled 3599
Index 3600
MatchType 3600
TKeyTipMatchItem Members 3598
TKeyTipMatchItem properties 3599
TKeyTipValidator class 3600
about TKeyTipValidator class 3600
TKeyTipValidator Members 3601
TKeyTipValidator methods 3601
Validate 3601
TLastButton class 1937
about TLastButton class 1937
Create 1939
TLastButton Members 1938
TLayoutGroup class 1939
about TLayoutGroup class 1939
Custom 1941
DisplayColumns 1941
Style 1942
StyleRule 1942
TLayoutGroup Members 1940
TLayoutGroup properties 1941
TLicenseInfo class 466
about TLicenseInfo class 466
Create 467
Desc 467
Destroy 467
Id 467
Key 468
LicensedUsers 468
TLicenseInfo fields 467
TLicenseInfo Members 466
TLicenseMaskInfo class 468
about TLicenseMaskInfo class 468
CapabilityMask 469
LicenseMask 469
TLicenseMaskInfo fields 469
TLicenseMaskInfo Members 469
TLimboTransactionInfo class 469
about TLimboTransactionInfo class 469
Action 471
Advise 471
HostSite 471
ID 471
MultiDatabase 472
RemoteDatabasePath 472
RemoteSite 472
State 472
TLimboTransactionInfo fields 470
TLimboTransactionInfo Members 470
TListActionLink class 1037
about TListActionLink class 1037
AddItem 1039
AddItem method 1038
IsActiveLinked 1039
IsImagesLinked 1039
RefreshControl 1040

SetAction 1040
SetActive 1040
SetImages 1040
SetItemIndex 1041
TListActionLink Members 1038
TListActionLink methods 1038
TListControlItem class 1041
about TListControlItem class 1041
Assign 1044
Caption 1046
Changed 1044
Create 1042
Data 1047
FCaption 1043
FData 1043
FImageIndex 1043
GetDisplayName 1045
ImageIndex 1047
SetCaption 1045
SetData 1045
SetImageIndex 1046
TListControlItem fields 1042
TListControlItem Members 1041
TListControlItem methods 1043
TListControlItem properties 1046
TListControlItems class 1047
about TListControlItems class 1047
Add 1049
CaseSensitive 1052
CompareItems 1050
Create 1049
CustomSort 1050
Items 1052
OnCompare 1054
Sort 1051
SortType 1053
TListControlItems events 1054
TListControlItems Members 1048
TListControlItems methods 1049
TListControlItems properties 1052
Update 1051
TLocalAppServer class 3182
about TLocalAppServer class 3182
AS_ApplyUpdates 3185
AS_DataRequest 3185
AS_Execute 3185
AS_GetParams 3185
AS_GetProviderNames 3186
AS_GetRecords 3186
AS_RowRequest 3186
Create 3183
Create constructor 3183
Destroy 3184
GetIDsOfNames 3186
GetTypeInfo 3187
GetTypeInfoCount 3187
InterfaceSupportsErrorInfo 3187
Invoke 3187
SafeCallException 3188
TLocalAppServer Members 3182
TLocalAppServer methods 3184
TMainMenu class 1322
about TMainMenu class 1322
AutoHotkeys 1327
AutoLineReduction 1328
AutoMerge 1328
BiDiMode 1329
GetHandle 1324
GetOle2AcceleratorTable 1324
Images 1329
MenuChanged 1325
Merge 1325
OnChange 1331
OwnerDraw 1330
ParentBiDiMode 1330
PopulateOle2Menu 1326
SetOle2MenuHandle 1326
TMainMenu events 1330

TMainMenu Members 1322
TMainMenu methods 1323
TMainMenu properties 1327
Unmerge 1326
TMask class 1178
about TMask class 1178
Create 1179
Destroy 1180
Matches 1180
TMask Members 1179
TMask methods 1180
TMaskEdit class 1100
about TMaskEdit class 1100
Align 1104
Alignment 1107
Anchors 1108
AutoSelect 1108
AutoSize 1109
BevelEdges 1110
BevelInner 1110
BevelKind 1111
BevelOuter 1111
BevelWidth 1112
BiDiMode 1112
BorderStyle 1112
CharCase 1113
Color 1113
Constraints 1114
Ctl3D 1115
DoubleBuffered 1115
DragCursor 1117
DragKind 1117
DragMode 1118
EditMask 1119
Enabled 1122
Font 1123
ImeMode 1124
ImeName 1124
MaxLength 1125
OnChange 1142
OnClick 1142
OnDblClick 1144
OnDragDrop 1144
OnDragOver 1147
OnEndDock 1150
OnEndDrag 1151
OnEnter 1152
OnExit 1154
OnKeyDown 1158
OnKeyPress 1162
OnKeyUp 1163
OnMouseActivate 1165
OnMouseDown 1166
OnMouseEnter 1168
OnMouseLeave 1168
OnMouseMove 1168
OnMouseUp 1171
OnStartDock 1173
OnStartDrag 1173
ParentBiDiMode 1125
ParentColor 1126
ParentCtl3D 1126
ParentDoubleBuffered 1127
ParentFont 1127
ParentShowHint 1127
PasswordChar 1128
PopupMenu 1128
ReadOnly 1133
ShowHint 1134
TabOrder 1135
TabStop 1136
Text 1136
TextHint 1139
TMaskEdit events 1141
TMaskEdit Members 1101
TMaskEdit properties 1103
Visible 1139
TMediaPlayer class 2305

about TMediaPlayer class 2305
Anchors 2332
AutoButtonSet 2312
AutoEnable 2333
AutoOpen 2336
AutoRewind 2337
Back 2313
Capabilities 2337
Click 2313
Close 2314
ColoredButtons 2337
Constraints 2338
Create 2311
Destroy 2311
DeviceID 2338
DeviceType 2339
Display 2339
DisplayRect 2340
DoNotify 2315
DoubleBuffered 2340
Eject 2315
Enabled 2342
EnabledButtons 2343
EndPos 2345
Error 2346
ErrorMessage 2346
FileName 2347
Frames 2349
KeyDown 2317
Length 2350
Loaded 2317
MMNotify 2318
Mode 2350
Next 2318
Notification 2319
Notify 2350
NotifyValue 2351
OnClick 2366
OnContextPopup 2367
OnEnter 2368
OnExit 2370
OnMouseEnter 2374
OnMouseLeave 2374
OnNotify 2375
OnPostClick 2375
Open 2319
Paint 2322
ParentDoubleBuffered 2352
ParentShowHint 2352
Pause 2322
PauseOnly 2323
Play 2323
PopupMenu 2352
Position 2357
PostClick 2326
Previous 2327
Resume 2327
Rewind 2328
Save 2328
Shareable 2358
ShowHint 2358
Start 2360
StartPos 2360
StartRecording 2329
Step 2330
Stop 2330
TabOrder 2361
TabStop 2361
TimeFormat 2362
TMediaPlayer events 2366
TMediaPlayer Members 2309
TMediaPlayer methods 2312
TMediaPlayer properties 2331
TrackLength 2362
TrackPosition 2362
Tracks 2363
Updated 2331
Visible 2363

VisibleButtons 2365
Wait 2365
TMemIniFile class 765
about TMemIniFile class 765
CaseSensitive 782
Clear 771
Create 770
Create constructor 770
DeleteKey 771
Destroy 770
Encoding 782
EraseSection 772
GetStrings 772
ReadSection 773
ReadSections 773
ReadSectionValues 774
ReadString 774
Rename 777
SetStrings 778
TMemIniFile Members 769
TMemIniFile methods 771
TMemIniFile properties 782
UpdateFile 778
WriteString 779
TMenu class 1331
about TMenu class 1331
AdjustBiDiBehavior 1335
AutoHotkeys 1343
AutoLineReduction 1344
BiDiMode 1344
Create 1333
CreateMenuItem 1335
Destroy 1334
DispatchCommand 1335
DispatchPopup 1336
DoBiDiModeChanged 1336
DoChange 1336
DoGetMenuString 1337
FindItem 1337
GetChildren 1338
GetHandle 1338
GetHelpContext 1338
Handle 1345
Images 1345
IsOwnerDraw 1339
IsRightToLeft 1339
IsShortCut 1339
Items 1346
Loaded 1340
MenuChanged 1340
Notification 1341
OnChange 1352
OwnerDraw 1350
ParentBiDiMode 1351
ParentBiDiModeChanged 1341
ParentBiDiModeChanged method 1341
ProcessMenuChar 1342
SetChildOrder 1342
TMenu events 1351
TMenu Members 1332
TMenu methods 1334
TMenu properties 1343
UpdateItems 1343
WindowHandle 1351
TMenuActionLink class 1352
about TMenuActionLink class 1352
AssignClient 1354
FClient 1354
IsAutoCheckLinked 1355
IsCaptionLinked 1355
IsCheckedLinked 1356
IsEnabledLinked 1356
IsGroupIndexLinked 1356
IsHelpContextLinked 1357
IsHintLinked 1357
IsImageIndexLinked 1358
IsOnExecuteLinked 1358
IsShortCutLinked 1358

IsVisibleLinked 1359
SetAutoCheck 1359
SetCaption 1359
SetChecked 1360
SetEnabled 1360
SetHelpContext 1361
SetHint 1361
SetImageIndex 1361
SetOnExecute 1362
SetShortCut 1362
SetVisible 1362
TMenuActionLink fields 1353
TMenuActionLink Members 1353
TMenuActionLink methods 1354
TMenuItem class 1363
about TMenuItem class 1363
Action 1393
ActionChange 1369
ActionLink 1394
Add 1369
Add method 1369
AdvancedDrawItem 1374
AssignTo 1374
AutoCheck 1394
AutoHotkeys 1394
AutoLineReduction 1395
Bitmap 1395
Break 1396
Caption 1396
Checked 1401
Clear 1375
Click 1375
Command 1401
Count 1402
Create 1366
Default 1406
DefineProperties 1375
Delete 1376
Destroy 1367
DoDrawText 1376
DrawItem 1377
Enabled 1406
Find 1377
GetActionLinkClass 1377
GetAutoHotkeys 1378
GetAutoLineReduction 1378
GetChildren 1378
GetCount 1378
GetEnumerator 1379
GetHandle 1379
GetImageList 1379
GetItem 1380
GetMenuIndex 1380
GetParentComponent 1380
GetParentMenu 1380
GroupIndex 1408
Handle 1409
HasParent 1381
HelpContext 1410
Hint 1410
ImageIndex 1411
IndexOf 1381
InitiateAction 1382
Insert 1382
InsertNewLine 1383
InsertNewLineAfter 1383
InsertNewLineBefore 1384
IsLine 1385
Items 1411
Loaded 1385
MeasureItem 1386
MenuChanged 1386
MenuIndex 1413
NewBottomLine 1386
NewTopLine 1387
Notification 1387
OnAdvancedDrawItem 1416
OnClick 1417

OnDrawItem 1422
OnMeasureItem 1423
Parent 1414
RadioItem 1414
Remove 1388
RethinkHotkeys 1388
RethinkLines 1389
SetBreak 1389
SetCaption 1389
SetChecked 1390
SetChildOrder 1390
SetDefault 1390
SetEnabled 1390
SetGroupIndex 1391
SetImageIndex 1391
SetMenuIndex 1391
SetParentComponent 1391
SetRadioItem 1392
SetShortCut 1392
SetVisible 1392
ShortCut 1415
SubMenuImages 1415
TMenuItem events 1416
TMenuItem Members 1363
TMenuItem methods 1368
TMenuItem properties 1393
UpdateItems 1392
Visible 1416
TMenuItemEnumerator class 1424
about TMenuItemEnumerator class 1424
Create 1424
Current 1425
GetCurrent 1425
MoveNext 1425
TMenuItemEnumerator Members 1424
TMenuItemEnumerator methods 1424
TMenuItemEnumerator properties 1425
TMenuItemStack class 1426
about TMenuItemStack class 1426
ClearItem 1426
TMenuItemStack Members 1426
TMenuItemStack methods 1426
TMidasConnection class 1782
about TMidasConnection class 1782
ConnectType 1786
Create 1783
DoConnect 1784
DoDisconnect 1785
ServerPort 1786
SetConnectType 1785
SetUseBroker 1785
TMidasConnection Members 1782
TMidasConnection methods 1784
TMidasConnection properties 1785
UseBroker 1787
TMidasPageProducer class 2295
about TMidasPageProducer class 2295
TMoreCommandsEvent
TCustomRibbonQuickAccessToolbar class 3453
TMtsAutoObject class 2388
about TMtsAutoObject class 2388
Activate 2389
CanBePooled 2390
Deactivate 2390
DisableCommit 2390
EnableCommit 2391
IsCallerInRole 2391
IsInTransaction 2392
IsSecurityEnabled 2392
ObjectContext 2394
OnActivate 2393
OnDeactivate 2393
Pooled 2394
SetAbort 2393
SetComplete 2393
TMtsAutoObject Members 2388
TMtsAutoObject methods 2389
TMtsAutoObject properties 2394

TMtsDataModule class 2398
about TMtsDataModule class 2398
Activate 2400
AS_ApplyUpdates 2401
AS_DataRequest 2401
AS_Execute 2402
AS_GetParams 2403
AS_GetProviderNames 2404
AS_GetRecords 2404
AS_RowRequest 2405
AutoComplete 2411
CallComplete 2406
CanBePooled 2406
Create 2399
Deactivate 2406
DisableCommit 2407
EnableCommit 2407
IsCallerInRole 2408
IsInTransaction 2408
IsSecurityEnabled 2409
ObjectContext 2411
OnActivate 2412
OnDeactivate 2412
Pooled 2412
SetAbort 2409
SetComplete 2410
TMtsDataModule events 2412
TMtsDataModule Members 2399
TMtsDataModule methods 2400
TMtsDataModule properties 2410
TMultipartContentParser class 3342
about TMultipartContentParser class 3342
BufferContent 3344
CanParse 3344
Destroy 3343
ExtractContentTypeFields 3344
ExtractMultipartContentFields 3345
GetContentFields 3345
GetFiles 3345
ParseMultiPart 3346
ParseMultiPartContent 3346
ParseMultipartHeaders 3346
TMultipartContentParser Members 3342
TMultipartContentParser methods 3343
TNextButton class 1943
about TNextButton class 1943
Create 1944
TNextButton Members 1943
TNextPageButton class 1945
about TNextPageButton class 1945
Create 1947
TNextPageButton Members 1945
TObjectDispatch class 2484
about TObjectDispatch class 2484
Create 2485
Destroy 2485
GetIDsOfNames 2486
GetMethodInfo 2486
GetObjectDispatch 2486
GetPropInfo 2487
GetTypeInfo 2487
GetTypeInfoCount 2487
Instance 2488
Invoke 2488
TObjectDispatch Members 2484
TObjectDispatch methods 2485
TObjectDispatch properties 2488
TOffice2007Button class 3461
about TOffice2007Button class 3461
CMMouseenter 3463
CMMouseleave 3463
Create 3462
Destroy 3462
DoClick 3463
DrawBackground 3464
DrawImage 3464
MouseDown 3464
MouseInControl 3467

MouseMove 3465
MouseUp 3465
OnClick 3467
Paint 3466
RibbonStyle 3467
SetParent 3466
TOffice2007Button events 3467
TOffice2007Button Members 3461
TOffice2007Button methods 3463
TOffice2007Button properties 3466
TogglePosition
TQuickAccessToolbar class 3473
TOleContainer class 2529
_AddRef 2534
_Release 2535
about TOleContainer class 2529
Align 2552
AllowActiveDoc 2556
AllowInPlace 2556
Anchors 2557
AutoActivate 2557
AutoVerbMenu 2558
BorderStyle 2558
CanPaste 2558
Caption 2559
Changed 2535
ChangeIconDialog 2535
Close 2536
Color 2561
Constraints 2562
Copy 2536
CopyOnSave 2563
Create 2532
CreateLinkToFile 2537
CreateObject 2537
CreateObjectFromFile 2538
CreateObjectFromInfo 2539
CreateParams 2540
Ctl3D 2563
DblClick 2540
DefineProperties 2541
Destroy 2533
DestroyObject 2541
DoEnter 2542
DoVerb 2542
DragCursor 2564
DragMode 2564
Enabled 2566
GetIconMetaPict 2542
GetPopupMenu 2543
Iconic 2566
InsertObjectDialog 2543
KeyDown 2543
Linked 2567
LoadFromFile 2544
LoadFromStream 2545
Modified 2567
MouseDown 2545
NewInserted 2567
ObjectPropertiesDialog 2546
ObjectVerbs 2568
OldStreamFormat 2568
OleClassName 2569
OleObject 2569
OleObjectInterface 2569
OnActivate 2583
OnDeactivate 2583
OnDragDrop 2584
OnDragOver 2587
OnEndDrag 2590
OnEnter 2591
OnExit 2593
OnKeyDown 2597
OnKeyPress 2601
OnKeyUp 2602
OnMouseActivate 2604
OnMouseDown 2605
OnMouseMove 2607

OnMouseUp 2609
OnObjectMove 2611
OnResize 2612
OnStartDrag 2612
Paint 2546
ParentColor 2570
ParentCtl3D 2570
ParentShowHint 2571
Paste 2546
PasteSpecialDialog 2547
PopupMenu 2571
PrimaryVerb 2576
QueryInterface 2547
Run 2548
SaveAsDocument 2548
SaveToFile 2550
SaveToStream 2550
ShowHint 2577
SizeMode 2578
SourceDoc 2579
State 2579
StorageInterface 2579
TabOrder 2580
TabStop 2580
TOleContainer events 2582
TOleContainer Members 2530
TOleContainer methods 2533
TOleContainer properties 2551
UpdateObject 2551
UpdateVerbs 2551
Visible 2581
TOleControl class 2638
_AddRef 2646
_Release 2647
_SetColorProp 2647
_SetTColorProp 2648
_SetTFontProp 2648
_SetTOleEnumProp 2649
_SetTPictureProp 2649
about TOleControl class 2638
Anchors 2695
BrowseProperties 2649
CanInPlaceActivate 2650
ContextSensitiveHelp 2650
ControlData 2696
Create 2643
CreateWnd 2651
DeactivateAndUndo 2651
DefaultDispatch 2696
DefaultHandler 2651
DefineProperties 2652
Destroy 2643
DestroyWindowHandle 2652
DiscardUndoState 2653
DoObjectVerb 2653
EnableModeless 2653
GetBorder 2654
GetByteProp 2654
GetColorProp 2655
GetCompProp 2655
GetContainer 2655
GetCurrencyProp 2656
GetDoubleProp 2656
GetEnumPropDesc 2656
GetExtendedControl 2657
GetHelpContext 2657
GetIDispatchProp 2657
GetIDsOfNames 2658
GetIntegerProp 2658
GetIUnknownProp 2658
GetMoniker 2659
GetObjectVerbs 2659
GetOleBoolProp 2659
GetOleDateProp 2660
GetOleEnumProp 2660
GetOleVariantProp 2660
GetPropDisplayString 2661
GetPropDisplayStrings 2661

GetProperty 2661
GetShortIntProp 2662
GetSingleProp 2662
GetSmallintProp 2662
GetStringProp 2663
GetTColorProp 2663
GetTDateTimeProp 2664
GetTFontProp 2664
GetTOleEnumProp 2664
GetTPictureProp 2665
GetTypeInfo 2665
GetTypeInfoCount 2665
GetVariantProp 2666
GetWideStringProp 2666
GetWindowContext 2667
GetWordBoolProp 2667
GetWordProp 2667
InitControlData 2668
InitControlInterface 2668
InsertMenus 2668
Invoke 2668
InvokeEvent 2669
InvokeMethod 2669
IsCustomProperty 2669
IsPropPageProperty 2669
LockInPlaceActive 2670
OleControlSite_TranslateAccelerator 2670
OleInPlaceFrame_GetWindow 2670
OleInPlaceFrame_TranslateAccelerator 2670
OleInPlaceSite_GetWindow 2671
OleObject 2697
OnChanged 2671
OnControlInfoChanged 2671
OnFocus 2671
OnInPlaceActivate 2672
OnInPlaceDeactivate 2672
OnPosRectChange 2672
OnRequestEdit 2672
OnShowWindow 2672
OnUIActivate 2673
OnUIDeactivate 2673
PaletteChanged 2673
PerPropBrowsing 2697
PictureChanged 2674
PostMessageFilter 2674
PreMessageFilter 2674
QueryInterface 2675
QueryService 2675
RemoveMenus 2675
RequestBorderSpace 2675
RequestNewObjectLayout 2676
SaveObject 2676
Scroll 2676
ServiceQuery 2698
SetActiveObject 2677
SetBorderSpace 2677
SetBounds 2677
SetByteProp 2677
SetColorProp 2678
SetCompProp 2678
SetCurrencyProp 2679
SetDoubleProp 2679
SetIDispatchProp 2679
SetIntegerProp 2680
SetIUnknownProp 2680
SetMenu 2680
SetName 2681
SetOleBoolProp 2681
SetOleDateProp 2681
SetOleEnumProp 2682
SetOleVariantProp 2682
SetParent 2682
SetPropDisplayString 2683
SetProperty 2683
SetShortIntProp 2684
SetSingleProp 2684
SetSmallintProp 2684
SetStatusText 2685

SetStringProp 2685
SetTColorProp 2685
SetTDateTimeProp 2686
SetTFontProp 2686
SetTOleEnumProp 2686
SetTPictureProp 2687
SetVariantProp 2687
SetWideStringProp 2688
SetWordBoolProp 2688
SetWordProp 2688
ShowAboutBox 2689
ShowObject 2689
ShowPropertyFrame 2689
StandardEvent 2690
SuppressException 2690
TabStop 2697
TOleControl events 2698
TOleControl Members 2639
TOleControl methods 2643
TOleControl properties 2695
TransformCoords 2690
WndProc 2691
TOleForm class 2613
about TOleForm class 2613
Create 2614
Destroy 2614
TOleForm Members 2613
TOLEnterpriseConnection class 1305
about TOLEnterpriseConnection class 1305
BrokerName 1308
ComputerName 1308
DoConnect 1307
TOLEnterpriseConnection Members 1306
TOLEnterpriseConnection methods 1307
TOLEnterpriseConnection properties 1308
TOleServer class 2708
_AddRef 2711
_Release 2712
about TOleServer class 2708
AutoConnect 2717
Connect 2712
ConnectEvents 2713
ConnectKind 2718
Create 2709
Destroy 2710
Disconnect 2713
DisconnectEvents 2713
EventDispatch 2718
GetAutoConnect 2714
GetConnectKind 2714
GetServer 2714
InitServerData 2715
InvokeEvent 2715
Loaded 2716
QueryInterface 2716
RemoteMachineName 2719
ServerData 2719
SetAutoConnect 2716
SetConnectKind 2717
TOleServer Members 2708
TOleServer methods 2711
TOleServer properties 2717
Top
TRibbonApplicationMenuBar class 3957
TopItem
TOutlineNode class 2956
TOptionItem class 3934
about TOptionItem class 3934
Action 3936
Assign 3935
Caption 3937
GetDisplayName 3936
Hint 3937
KeyTip 3937
Tag 3937
TOptionItem Members 3934
TOptionItem methods 3935
TOptionItem properties 3936

TOptionItems class 3938
about TOptionItems class 3938
Add 3938
ApplicationMenu 3939
Insert 3939
Items 3940
TOptionItems Members 3938
TOptionItems methods 3938
TOptionItems properties 3939
TOPToSoapDomConvert class 2747
about TOPToSoapDomConvert class 2747
Attachments 2756
Create 2751
Destroy 2751
Encoding 2756
InvContextToMsg 2752
MakeFault 2752
MakeResponse 2753
MsgToInvContext 2754
MsgToInvContext method 2753
NewXMLDocument 2754
ProcessResponse 2755
ProcessResponse method 2755
TempDir 2757
TOPToSoapDomConvert Members 2748
TOPToSoapDomConvert methods 2751
TOPToSoapDomConvert properties 2756
WSDLView 2758
TOutline class 2874
about TOutline class 2874
Align 2878
BorderStyle 2881
Color 2881
Ctl3D 2882
DoubleBuffered 2883
DragCursor 2885
DragKind 2885
DragMode 2885
Enabled 2887
Font 2887
ItemHeight 2888
ItemSeparator 2889
Lines 2889
OnClick 2906
OnCollapse 2908
OnContextPopup 2908
OnDblClick 2909
OnDragDrop 2909
OnDragOver 2912
OnDrawItem 2916
OnEndDock 2916
OnEndDrag 2917
OnEnter 2918
OnExit 2920
OnExpand 2924
OnKeyDown 2924
OnKeyPress 2928
OnKeyUp 2929
OnMouseActivate 2931
OnMouseDown 2932
OnMouseEnter 2934
OnMouseLeave 2934
OnMouseMove 2935
OnMouseUp 2937
OnStartDock 2939
OnStartDrag 2940
Options 2890
OutlineStyle 2890
ParentColor 2891
ParentCtl3D 2892
ParentDoubleBuffered 2892
ParentFont 2892
ParentShowHint 2893
PictureClosed 2893
PictureLeaf 2894
PictureMinus 2894
PictureOpen 2895
PicturePlus 2895

PopupMenu 2896
ScrollBars 2901
ShowHint 2901
Style 2902
TabOrder 2903
TabStop 2904
TOutline events 2905
TOutline Members 2874
TOutline properties 2877
Visible 2904
TOutlineNode class 2940
about TOutlineNode class 2940
AddNode 2944
ChangeLevelBy 2944
Collapse 2944
Create 2942
Data 2951
Destroy 2943
Expand 2945
ExpandCount 2952
Expanded 2952
FullExpand 2945
FullPath 2952
GetDataItem 2946
GetDisplayWidth 2946
getFirstChild 2946
GetLastChild 2947
GetNextChild 2947
GetNodeAtIndex 2947
GetPrevChild 2948
GetRowOfNode 2948
GetTextItem 2948
GetVisibleNode 2949
HasAsParent 2949
HasItems 2953
Index 2953
InsertNode 2949
InternalRemove 2949
IsVisible 2954
Items 2954
Level 2954
List 2955
MoveTo 2949
Outline 2955
Parent 2955
Remove 2950
Text 2955
TopItem 2956
TOutlineNode Members 2941
TOutlineNode methods 2943
TOutlineNode properties 2951
WriteNode 2950, 2951
WriteNode method 2950
TPacketDataSet class 3188
about TPacketDataSet class 3188
AssignCurValues 3190
AssignCurValues method 3190
Create 3189
CreateFromDelta 3191
DataEvent 3191
GetStateFieldValue 3191
HasCurValues 3192
HasMergeConflicts 3192
InitAltRecBuffers 3192
InternalClose 3193
InternalInitRecord 3193
InternalOpen 3193
NewValuesModified 3195
SetFieldData 3194
SetStateFieldValue 3194
StreamMetaData 3195
TPacketDataSet Members 3189
TPacketDataSet methods 3190
TPacketDataSet properties 3194
UpdateKind 3194
UseCurValues 3195
WritingCurValues 3196
TPageItemsProducer class 2970

about TPageItemsProducer class 2970
Create 2971
Destroy 2971
GetTemplateStream 2972
HTMLDoc 2972
HTMLFile 2973
TPageItemsProducer Members 2971
TPageItemsProducer methods 2972
TPageItemsProducer properties 2972
TPanningWindow class 693
about TPanningWindow class 693
Create 694
CreateParams 695
Destroy 694
DoUpdate 696
GetIsPanning 696
GetPanOptions 696
IsPanning 703
MouseMove 697
OnUpdate 704
Paint 697
PanControl 703
PanInterval 704
PanOptions 704
SetPanCursor 698
StartPanning 698
StopPanning 698
TPanningWindow events 704
TPanningWindow Members 693
TPanningWindow methods 695
TPanningWindow properties 703
UpdateScroll 698
WndProc 699
TPB
TIBTransaction class 219
TPBLength
TIBTransaction class 219
TPlatformDefaultStyleActionBars class 2976
about TPlatformDefaultStyleActionBars class 2976
GetAddRemoveItemClass 2977
GetColorMapClass 2977
GetControlClass 2977
GetPopupClass 2978
GetScrollBtnClass 2978
GetStyleName 2978
TPlatformDefaultStyleActionBars Members 2976
TPlatformDefaultStyleActionBars methods 2976
TPngImage class 2996
about TPngImage class 2996
TPNGList class 2997
about TPNGList class 2997
Add 2998
FindChunk 2998
Item 2999
ItemFromClass 2998
RemoveChunk 2999
TPNGList Members 2997
TPNGList methods 2998
TPNGList properties 2999
TPNGPointerList class 2999
about TPNGPointerList class 2999
Add 3001
Count 3002
Create 3000
Destroy 3000
Insert 3001
Item 3002
Owner 3002
Remove 3001
SetSize 3001
TPNGPointerList Members 3000
TPNGPointerList methods 3001
TPNGPointerList properties 3002
TPopupList class 1427
about TPopupList class 1427
Add 1428
FWindow 1428
MainWndProc 1429

Remove 1429
TPopupList fields 1428
TPopupList Members 1427
TPopupList methods 1428
TPopupList properties 1430
Window 1430
WndProc 1429
TPopupMenu class 1430
about TPopupMenu class 1430
Alignment 1436
AutoHotkeys 1436
AutoLineReduction 1437
AutoPopup 1437
BiDiMode 1438
Create 1432
Destroy 1433
DoPopup 1433
HelpContext 1438
Images 1439
MenuAnimation 1439
OnChange 1443
OnPopup 1444
OwnerDraw 1440
ParentBiDiMode 1440
Popup 1434
PopupComponent 1440
PopupPoint 1443
SetPopupPoint 1434
TPopupMenu events 1443
TPopupMenu Members 1431
TPopupMenu methods 1433
TPopupMenu properties 1435
TrackButton 1443
UseRightToLeftAlignment 1435
TPostButton class 1947
about TPostButton class 1947
Create 1949
TPostButton Members 1948
TPrinter class 3038
Abort 3041
Aborted 3051
about TPrinter class 3038
BeginDoc 3043
Canvas 3053
Capabilities 3053
Copies 3054
Create 3040
Destroy 3040
EndDoc 3045
Fonts 3054
GetPrinter 3047
Handle 3054
NewPage 3047
Orientation 3056
PageHeight 3057
PageNumber 3059
PageWidth 3060
PrinterIndex 3062
Printers 3062
Printing 3062
Refresh 3049
SetPrinter 3050
Title 3065
TPrinter Members 3039
TPrinter methods 3040
TPrinter properties 3050
TPriorButton class 1949
about TPriorButton class 1949
Create 1951
TPriorButton Members 1950
TPriorPageButton class 1951
about TPriorPageButton class 1951
Create 1953
TPriorPageButton Members 1952
TProvider class 3196
about TProvider class 3196
TQueryButton class 1953
about TQueryButton class 1953

Caption 1959
Create 1955
Custom 1960
DefaultCaption 1955
EventContent 1956
GetCaption 1956
GetHTMLControlName 1957
GetHTMLForm 1957
GetInputType 1957
GetXMLComponent 1957
ImplContent 1958
InputType 1956
Notification 1958
ParentXMLComponent 1959
SetXMLComponent 1959
SetXMLUseParent 1959
Style 1960
StyleRule 1961
TQueryButton fields 1955
TQueryButton Members 1954
TQueryButton methods 1956
TQueryButton properties 1959
XMLComponent 1961
XMLUseParent 1962
TQueryButtons class 1962
about TQueryButtons class 1962
Custom 1964
Style 1964
StyleRule 1965
TQueryButtons Members 1963
TQueryButtons properties 1964
TQueryFieldGroup class 1965
about TQueryFieldGroup class 1965
Custom 1970
FindParam 1969
GetParamsList 1969
ImplCanAddClass 1969
IsParamInUse 1970
Style 1971
StyleRule 1971
TQueryFieldGroup Members 1967
TQueryFieldGroup methods 1969
TQueryFieldGroup properties 1970
TQueryForm class 1971
about TQueryForm class 1971
Action 1974
Custom 1974
Method 1975
Style 1975
StyleRule 1975
TQueryForm Members 1972
TQueryForm properties 1973
TQueryRadioGroup class 1976
about TQueryRadioGroup class 1976
AddAttributes 1979
Caption 1981
CaptionAttributes 1981
CaptionPosition 1982
Custom 1982
DataSet 1982
DisplayColumns 1983
DisplayWidth 1983
FieldName 1984
GetCheckIndex 1979
GetText 1980
IsQueryField 1980
Items 1984
ItemsField 1985
ParamName 1985
ReadOnly 1986
SetText 1980
Style 1986
StyleRule 1986
TabIndex 1987
Text 1987
TQueryRadioGroup Members 1977
TQueryRadioGroup methods 1979
TQueryRadioGroup properties 1980

Values 1988
ValuesField 1988
TQuerySelectOptions class 1989
about TQuerySelectOptions class 1989
Caption 1993
CaptionAttributes 1994
CaptionPosition 1994
Custom 1994
DataSet 1995
DisplayRows 1995
FieldName 1996
GetSelectIndex 1992
GetText 1992
IsQueryField 1992
Items 1996
ItemsField 1997
ParamName 1997
SetText 1993
Style 1998
StyleRule 1998
TabIndex 1999
Text 1999
TQuerySelectOptions Members 1989
TQuerySelectOptions methods 1992
TQuerySelectOptions properties 1993
Values 1999
ValuesField 2000
TQueryText class 2000
about TQueryText class 2000
AddAttributes 2004
Caption 2005
CaptionAttributes 2005
CaptionPosition 2006
Create 2003
Custom 2006
DisplayWidth 2007
FieldName 2007
GetText 2004
IsQueryField 2004
MaxWidth 2007
ParamName 2008
ReadOnly 2008
SetText 2004
Style 2009
StyleRule 2009
TabIndex 2010
Text 2010
TQueryText Members 2001
TQueryText methods 2003
TQueryText properties 2005
TQueryTextArea class 2010
about TQueryTextArea class 2010
AddAttributes 2014
Caption 2016
CaptionAttributes 2016
CaptionPosition 2017
ControlContent 2014
Create 2013
Custom 2017
Destroy 2013
DisplayRows 2017
DisplayWidth 2018
FieldName 2018
GetText 2014
IsQueryField 2015
Lines 2019
ParamName 2019
ReadOnly 2020
SetLines 2015
SetText 2015
Style 2020
StyleRule 2020
TabIndex 2021
TQueryTextArea Members 2011
TQueryTextArea methods 2013
TQueryTextArea properties 2015
Wrap 2021
TQuickAccessToolbar class 3469

about TQuickAccessToolbar class 3469
ActionBar 3473
Assign 3472
BarWidth 3474
Create 3470
DoChange 3472
OnChange 3475
Position 3474
Show 3473
ShowMinimize 3474
ShowMoreCommands 3474
ShowPosition 3474
TogglePosition 3473
TQuickAccessToolbar enumerations 3471
TQuickAccessToolbar events 3475
TQuickAccessToolbar Members 3470
TQuickAccessToolbar methods 3471
TQuickAccessToolbar properties 3473
TQuickAccessToolbarGroupItem class 3475
about TQuickAccessToolbarGroupItem class 3475
Action 3476
Caption 3477
Create 3476
Destroy 3476
Group 3477
TQuickAccessToolbarGroupItem Members 3475
TQuickAccessToolbarGroupItem properties 3476
TQuickAccessToolbarGroups class 3477
about TQuickAccessToolbarGroups class 3477
Add 3478
Clean 3479
Create 3478
Destroy 3478
Exists 3479
Group 3479
Remove 3479
TQuickAccessToolbarGroups Members 3477
TQuickAccessToolbarGroups methods 3478
TraceFlags
IIBSQLMonitorHook interface 574
TIBCustomService class 432
TIBCustomSQLMonitor class 564
TIBDataBase class 197
TIBSQLMonitor class 566
TrackButton
TPopupMenu class 1443
TrackLength
TMediaPlayer class 2362
TrackMenu
TRibbonActionPopupMenu class 3942
TrackPosition
TMediaPlayer class 2362
Tracks
TMediaPlayer class 2363
Transaction
TIBBase class 174
TIBBlobStream class 53
TIBCustomDataSet class 106
TIBExtract class 282
TIBSQL class 535
TransactionCount
TIBDataBase class 198
TransactionFree
TIBCustomDataSet class 111
TIBDataSet class 134
TIBQuery class 395
TIBStoredProc class 596
TIBTable class 624
Transactions
TIBDataBase class 198
TransactionStarted
TDataSetProvider class 3163
TransformCoords
TOleControl class 2690
TranslateURI
TISAPIRequest class 975
Transliterate
TIBBatchMove class 37

TRCommit
IIBSQLMonitorHook interface 572
TRCommitRetaining
IIBSQLMonitorHook interface 572
TRegIniFile class 3237
about TRegIniFile class 3237
Create 3239
Create constructor 3239
DeleteKey 3240
EraseSection 3241
FileName 3246
ReadBool 3241
ReadInteger 3242
ReadSection 3242
ReadSections 3243
ReadSectionValues 3243
ReadString 3244
TRegIniFile Members 3237
TRegIniFile methods 3240
TRegIniFile properties 3246
WriteBool 3244
WriteInteger 3245
WriteString 3245
TRegistry class 3246
about TRegistry class 3246
Access 3293
ChangeKey 3255
CloseKey 3255
Create 3251
Create constructor 3251
CreateKey 3258
CurrentKey 3296
CurrentPath 3296
DeleteKey 3259
DeleteValue 3261
Destroy 3253
GetBaseKey 3262
GetData 3262
GetDataAsString 3262
GetDataInfo 3263
GetDataSize 3263
GetDataType 3264
GetKey 3264
GetKeyInfo 3264
GetKeyNames 3265
GetValueNames 3265
HasSubKeys 3266
KeyExists 3266
LazyWrite 3296
LoadKey 3268
MoveKey 3269
OpenKey 3270
OpenKeyReadOnly 3272
PutData 3273
ReadBinaryData 3273
ReadBool 3274
ReadCurrency 3274
ReadDate 3274
ReadDateTime 3275
ReadFloat 3275
ReadInteger 3276
ReadString 3276
ReadTime 3277
RegistryConnect 3277
RenameValue 3278
ReplaceKey 3278
RestoreKey 3279
RootKey 3297
SaveKey 3279
SetCurrentKey 3280
TRegistry Members 3249
TRegistry methods 3254
TRegistry properties 3292
UnLoadKey 3280
ValueExists 3281
WriteBinaryData 3281
WriteBool 3282
WriteCurrency 3282

WriteDate 3282
WriteDateTime 3283
WriteExpandString 3286
WriteFloat 3288
WriteInteger 3289
WriteString 3289
WriteTime 3292
TRegistryClass class 2511
about TRegistryClass class 2511
Create 2512
Destroy 2512
TRegistryClass Members 2511
TRegistryClass methods 2512
UpdateRegistry 2512
TRegistryIniFile class 3299
about TRegistryIniFile class 3299
Create 3304
Create constructor 3304
DeleteKey 3305
Destroy 3304
EraseSection 3306
ReadBinaryStream 3306
ReadDate 3307
ReadDateTime 3307
ReadFloat 3308
ReadInteger 3308
ReadSection 3311
ReadSections 3312
ReadSections method 3312
ReadSectionValues 3313
ReadString 3313
ReadTime 3316
RegIniFile 3326
TRegistryIniFile Members 3303
TRegistryIniFile methods 3305
TRegistryIniFile properties 3326
UpdateFile 3317
WriteBinaryStream 3317
WriteDate 3317
WriteDateTime 3318
WriteFloat 3319
WriteInteger 3319
WriteString 3322
WriteTime 3325
TRemotable class 868
about TRemotable class 868
Create 870
DataContext 873
Destroy 870
FSerializationOptions 871
ObjectToSOAP 871
SerializationOptions 873
SOAPToObject 872
TRemotable fields 871
TRemotable Members 869
TRemotable methods 871
TRemotable properties 873
TRemotableTypeRegistry class 874
about TRemotableTypeRegistry class 874
AutoRegisterNativeTypes 891
ClassOptions 877
ClassToURI 878
ClassToURI method 878
Create 876
Destroy 876
GetExternalPropName 879
GetInternalPropName 879
GetURICount 879
GetURIMap 880
GetURIMap method 880
GetVarTypeFromXSD 881
GetXSDInfoForClass 881
InfoToURI 882
IsClassScalar 882
Lock 883
RegisterExternalPropName 883
RegisterHolderClsMember 883
RegisterSerializeOptions 884

RegisterSerializeOptions method 884
RegisterXSClass 885
RegisterXSInfo 886
SerializeOptions 887
SerializeOptions method 887
TRemotableTypeRegistry Members 875
TRemotableTypeRegistry methods 876
TRemotableTypeRegistry properties 891
TypeInfoToXSD 887
UnLock 888
UnRegisterXSClass 888
UnRegisterXSInfo 889
URIToClass 889
URIToClass method 889
URIToInfo 890
VariantToInfo 890
XSDToTypeInfo 891
TRemotableXS class 892
about TRemotableXS class 892
NativeToXS 893
ObjectToSOAP 894
SOAPToObject 894
TRemotableXS Members 893
TRemotableXS methods 893
XSToNative 895
TRemoteServer class 1787
about TRemoteServer class 1787
TResetQueryButton class 2022
about TResetQueryButton class 2022
Create 2023
EventContent 2024
TResetQueryButton Members 2022
TResetQueryButton methods 2023
TResourceStream class 3003
about TResourceStream class 3003
Create 3003
Read 3004
Seek 3004
SetSize 3004
TResourceStream Members 3003
TResourceStream methods 3004
Write 3005
TRHandle
TIBBase class 174
TIBBlobStream class 53
TIBCustomDataSet class 106
TIBSQL class 535
TRibbon class 3480
about TRibbon class 3480
ActionManager 3484
Align 3484
Anchors 3488
ApplicationMenu 3488
BiDiMode 3488
Caption 3489
DocumentName 3492
Enabled 3492
Font 3492
Height 3493
HideTabs 3495
OnHelpButtonClick 3498
OnRecentItemClick 3498
OnTabChange 3498
ParentBiDiMode 3495
ParentFont 3495
QuickAccessToolbar 3496
ScreenTips 3496
ShowHelpButton 3496
Style 3497
TabIndex 3497
Tabs 3497
TRibbon events 3498
TRibbon Members 3480
TRibbon properties 3483
UseCustomFrame 3497
TRibbonActionPopupMenu class 3940
about TRibbonActionPopupMenu class 3940
CreateControl 3941

NCPaint 3941
SelectFirstItem 3942
TrackMenu 3942
TRibbonActionPopupMenu Members 3941
TRibbonActionPopupMenu methods 3941
TRibbonActionPopupMenu properties 3942
TRibbonAddRemoveItem class 3601
about TRibbonAddRemoveItem class 3601
DrawBackground 3602
DrawGlyph 3602
TRibbonAddRemoveItem Members 3602
TRibbonAddRemoveItem methods 3602
TRibbonApplicationButton class 3603
about TRibbonApplicationButton class 3603
CalcBounds 3603
TRibbonApplicationButton Members 3603
TRibbonApplicationButton methods 3603
TRibbonApplicationMenuBar class 3942
about TRibbonApplicationMenuBar class 3942
ActionManager 3950
Align 3950
CloseMenu 3945
Create 3944
CreateControl 3945
CreateControls 3946
Destroy 3944
DragOver 3946
DrawBackground 3946
GetControlClass 3946
GetRibbon 3947
Height 3954
HorzMargin 3955
Left 3955
NewPopup 3947
Notification 3947
OptionItems 3956
Paint 3948
RecentItems 3956
RibbonStyle 3948
SelectApplicationButton 3948
SetBounds 3948
SetRibbon 3949
ShowKeyTips 3949
Spacing 3956
Top 3957
TRibbonApplicationMenuBar Members 3943
TRibbonApplicationMenuBar methods 3945
TRibbonApplicationMenuBar properties 3949
VertMargin 3958
VisibleMenu 3959
VisibleMenuCount 3959
Width 3959
WMSysCommand 3949
TRibbonApplicationMenuButton class 3604
about TRibbonApplicationMenuButton class 3604
CalcBounds 3606
CMMouseEnter 3607
CMMouseLeave 3607
Create 3605
Destroy 3605
DragOver 3607
GetKeyTip 3607
HandleKeyTip 3608
HideKeyTip 3608
KeyTip 3612
KeyTipClick 3609
MouseInControl 3612
Paint 3609
RibbonStyle 3609
SetKeyTip 3610
SetParent 3610
ShowKeyTip 3611
TRibbonApplicationMenuButton Members 3604
TRibbonApplicationMenuButton methods 3606
TRibbonApplicationMenuButton properties 3611
UMFadeControl 3611
WMLButtonDblClk 3611
TRibbonApplicationMenuItem class 3612

about TRibbonApplicationMenuItem class 3612
CalcBounds 3614
Create 3613
DrawGlyph 3614
DrawSelectedBackground 3615
DrawSeparator 3615
DrawShadowedText 3615
DrawSubMenuGlyph 3615
DrawText 3616
GetImageSize 3616
TRibbonApplicationMenuItem Members 3613
TRibbonApplicationMenuItem methods 3614
TRibbonApplicationPopupMenu class 3961
about TRibbonApplicationPopupMenu class 3961
AlignControls 3963
CMTextchanged 3963
Create 3962
CreateControls 3963
CreateOptionsItems 3964
CreateRecentDocuments 3964
CreateWnd 3964
CustomAlignPosition 3965
Destroy 3962
DoDropActions 3965
DoDropCategory 3966
DoDropItem 3966
DrawBackground 3966
GetBorderHeight 3966
GetRibbon 3966
NCPaint 3967
PositionPopup 3967
RibbonStyle 3967
SetColorMap 3967
SetRibbon 3968
TRibbonApplicationPopupMenu Members 3961
TRibbonApplicationPopupMenu methods 3962
TRibbonBaseButtonControl class 3616
about TRibbonBaseButtonControl class 3616
AllowHotKeys 3619
CalcBounds 3619
CanAddToQuickAccessToolbar 3620
Click 3620
Create 3617
Destroy 3618
DragDrop 3620
DrawBackground 3621
DrawDown 3626
DrawGlyph 3621
DrawSeparator 3621
DrawShadowedText 3622
DrawText 3622
GetImageSize 3622
GetKeyTip 3622
HandleKeyTip 3623
HideKeyTip 3623
IsButton 3624
IsRibbonAvailable 3624
KeyTipClick 3624
RibbonStyle 3624
SetKeyTip 3625
ShowKeyTip 3625
TRibbonBaseButtonControl Members 3617
TRibbonBaseButtonControl methods 3619
TRibbonBaseButtonControl properties 3625
TRibbonCheckBoxControl class 3626
about TRibbonCheckBoxControl class 3626
CalcBounds 3627
DrawBackground 3628
DrawGlyph 3628
TRibbonCheckBoxControl Members 3626
TRibbonCheckBoxControl methods 3627
TRibbonComboBox class 3628
about TRibbonComboBox class 3628
Alignment 3631
AutoSelect 3631
CharCase 3632
Font 3632
HideSelection 3633

ImeMode 3634
ImeName 3634
Items 3635
MaxLength 3635
NumbersOnly 3636
OnChange 3648
OnClick 3649
OnContextPopup 3651
OnDblClick 3651
OnDragDrop 3652
OnDragOver 3655
OnEndDock 3658
OnEndDrag 3658
OnEnter 3659
OnExit 3662
OnKeyDown 3666
OnKeyPress 3669
OnKeyUp 3670
OnMouseActivate 3672
OnMouseDown 3673
OnMouseEnter 3675
OnMouseLeave 3676
OnMouseMove 3676
OnMouseUp 3678
OnStartDock 3680
OnStartDrag 3681
ParentFont 3636
PasswordChar 3636
PopupMenu 3637
ReadOnly 3642
ShowHint 3642
TabOrder 3644
TabStop 3644
Text 3645
TextHint 3646
TRibbonComboBox events 3648
TRibbonComboBox Members 3629
TRibbonComboBox properties 3630
Visible 3646
TRibbonComboBoxItem class 3682
about TRibbonComboBoxItem class 3682
CalcBounds 3683
DrawBackground 3683
DrawText 3683
TRibbonComboBoxItem Members 3682
TRibbonComboBoxItem methods 3682
TRibbonComboControl class 3684
about TRibbonComboControl class 3684
CalcBounds 3687
CanAddToQuickAccessToolbar 3687
Click 3688
CMBidimodechanged 3688
CMMouseenter 3688
CMMouseleave 3688
CMVisiblechanged 3689
ComboBox 3697
ComboClick 3689
ContainedControl 3697
Create 3685
Destroy 3686
DrawBackground 3689
DrawFrame 3689
DropDownClick 3690
GetCaptionWidth 3690
GetContainedControl 3690
GetGlyphWidth 3690
GetKeyTip 3690
GetLabelWidth 3691
HandleKeyTip 3691
HideKeyTip 3692
KeyTipClick 3692
MouseDown 3692
MouseInControl 3697
Notification 3693
Paint 3693
RibbonStyle 3694
SetActionClient 3694
SetBounds 3694

SetContainedControl 3694
SetDragMode 3695
SetKeyTip 3695
SetParent 3695
ShowComboItems 3696
ShowKeyTip 3696
TRibbonComboControl Members 3684
TRibbonComboControl methods 3686
TRibbonComboControl properties 3697
TRibbonCustomControl class 3698
about TRibbonCustomControl class 3698
CalcBounds 3700
CanAddToQuickAccessToolbar 3701
Click 3701
CMVisiblechanged 3701
ContainedControl 3710
Create 3699
Destroy 3699
DrawBackground 3702
GetCaptionWidth 3702
GetContainedControl 3702
GetGlyphWidth 3702
GetKeyTip 3703
GetLabelWidth 3703
HandleKeyTip 3703
HideKeyTip 3704
KeyTipClick 3704
Notification 3704
Paint 3705
SetActionClient 3705
SetBounds 3705
SetContainedControl 3708
SetDragMode 3708
SetKeyTip 3708
SetParent 3709
ShowKeyTip 3709
TRibbonCustomControl Members 3698
TRibbonCustomControl methods 3700
TRibbonCustomControl properties 3710
TRibbonCustomizeDlg class 3499
about TRibbonCustomizeDlg class 3499
SetupDlg 3499
TRibbonCustomizeDlg Members 3499
TRibbonCustomizeDlg methods 3499
TRibbonCustomizeItem class 3710
about TRibbonCustomizeItem class 3710
CalcBounds 3711
DrawBackground 3712
DrawGlyph 3712
RibbonStyle 3712
TRibbonCustomizeItem Members 3711
TRibbonCustomizeItem methods 3711
TRibbonCustomStatusControl class 3713
about TRibbonCustomStatusControl class 3713
CalcBounds 3714
DrawBackground 3715
TRibbonCustomStatusControl Members 3713
TRibbonCustomStatusControl methods 3714
TRibbonDropDownButton class 3715
about TRibbonDropDownButton class 3715
CalcBounds 3717
CalcLayout 3717
Click 3717
DoDropDown 3718
DrawArrow 3718
DrawBackground 3718
DrawShadowedText 3718
DrawText 3719
GetArrowPosition 3719
GetPopupClass 3719
KeyTipClick 3719
TRibbonDropDownButton Members 3715
TRibbonDropDownButton methods 3716
TRibbonGallery class 3720
about TRibbonGallery class 3720
TRibbonGalleryBar class 3990
about TRibbonGalleryBar class 3990
Items 3991

ItemsPerRow 3992
ShowItemCaptions 3992
TRibbonGalleryBar Members 3991
TRibbonGalleryBar properties 3991
TRibbonGalleryControl class 3720
about TRibbonGalleryControl class 3720
CanAddToQuickAccessToolbar 3721
ContainedControl 3722
Gallery 3722
GetContainedControl 3721
SetContainedControl 3721
TRibbonGalleryControl Members 3720
TRibbonGalleryControl methods 3721
TRibbonGalleryControl properties 3722
TRibbonGalleryControlMenuItem class 3722
about TRibbonGalleryControlMenuItem class 3722
CalcBounds 3723
CanAddToQuickAccessToolbar 3724
ContainedControl 3728
GetContainedControl 3724
Paint 3724
SetBounds 3725
SetContainedControl 3727
SetParent 3727
TRibbonGalleryControlMenuItem Members 3723
TRibbonGalleryControlMenuItem methods 3723
TRibbonGalleryControlMenuItem properties 3728
TRibbonGalleryItem class 3728
about TRibbonGalleryItem class 3728
CalcBounds 3730
CalcLayout 3730
TRibbonGalleryItem Members 3729
TRibbonGalleryItem methods 3730
TRibbonGalleryMenuItem class 3730
about TRibbonGalleryMenuItem class 3730
CalcBounds 3731
DrawBackground 3732
DrawGlyph 3732
DrawShadowedText 3732
DrawText 3733
TRibbonGalleryMenuItem Members 3731
TRibbonGalleryMenuItem methods 3731
TRibbonGalleryPopupBar class 3992
about TRibbonGalleryPopupBar class 3992
TRibbonGallerySplitButton class 3733
about TRibbonGallerySplitButton class 3733
GetPopupClass 3735
TRibbonGallerySplitButton Members 3733
TRibbonGallerySplitButton methods 3735
TRibbonGripItem class 3735
about TRibbonGripItem class 3735
CalcBounds 3737
Create 3736
GalleryResize 3738
Paint 3737
RibbonStyle 3738
TRibbonGripItem Members 3736
TRibbonGripItem methods 3737
TRibbonGripItem properties 3738
TRibbonGroup class 3500
about TRibbonGroup class 3500
ActionManager 3503
BiDiMode 3503
Caption 3503
DialogAction 3504
GroupAlign 3504
GroupIndex 3504
Height 3505
KeyTip 3505
ParentBiDiMode 3505
Rows 3505
TRibbonGroup Members 3501
TRibbonGroup properties 3503
TRibbonGroupCollapsedControl class 3506
about TRibbonGroupCollapsedControl class 3506
ActionManager 3515
AdjustBounds 3509
Caption 3515

Click 3509
CloseUp 3509
CMFontchanged 3510
CMMouseenter 3510
CMMouseleave 3510
CMTextchanged 3510
Create 3507
Destroy 3508
GetKeyTip 3510
GetParentControl 3511
GetTopPosition 3511
Group 3518
HandleKeyTip 3511
HideKeyTip 3512
KeyTip 3518
KeyTipClick 3512
MouseInControl 3518
Notification 3512
OldWidth 3518
Paint 3513
SetKeyTip 3513
SetupGroupImage 3514
ShowKeyTip 3514
TRibbonGroupCollapsedControl Members 3506
TRibbonGroupCollapsedControl methods 3508
TRibbonGroupCollapsedControl properties 3514
TRibbonGroupDockForm class 3519
about TRibbonGroupDockForm class 3519
Control 3521
Create 3519
CreateParams 3520
ParentControl 3521
TRibbonGroupDockForm Members 3519
TRibbonGroupDockForm methods 3520
TRibbonGroupDockForm properties 3520
WMActivate 3520
TRibbonGroupDropDown class 3738
about TRibbonGroupDropDown class 3738
CloseUp 3740
DoDropDown 3740
DrawArrow 3741
DrawBackground 3741
GetParentControl 3741
GetTopPosition 3741
Group 3742
TRibbonGroupDropDown Members 3739
TRibbonGroupDropDown methods 3740
TRibbonGroupDropDown properties 3742
TRibbonGroupScrollButton class 3521
about TRibbonGroupScrollButton class 3521
CMDesignHitTest 3523
CMMouseenter 3523
CMMouseleave 3523
Create 3522
Direction 3524
MouseInControl 3524
Paint 3523
RibbonStyle 3524
TRibbonGroupScrollButton Members 3521
TRibbonGroupScrollButton methods 3522
TRibbonGroupScrollButton properties 3524
TRibbonLargeButtonControl class 3742
about TRibbonLargeButtonControl class 3742
CalcBounds 3744
CalcLayout 3744
DrawBackground 3744
DrawShadowedText 3745
DrawText 3745
TRibbonLargeButtonControl Members 3742
TRibbonLargeButtonControl methods 3743
TRibbonLargeSplitButton class 3745
about TRibbonLargeSplitButton class 3745
CalcBounds 3747
CalcLayout 3748
Click 3748
CMMouseleave 3748
CursorInSplit 3752
DoSplitButtonClick 3748

DrawArrow 3749
DrawBackground 3749
GetArrowPosition 3749
InSplit 3749
KeyTipClick 3750
MouseDown 3750
MouseMove 3750
MouseUp 3751
TRibbonLargeSplitButton Members 3746
TRibbonLargeSplitButton methods 3747
TRibbonLargeSplitButton properties 3752
UpdateSplitBounds 3751
TRibbonLunaColorMap class 3525
about TRibbonLunaColorMap class 3525
ActiveTabFontColor 3527
BevelMinimizedInner 3527
BevelMinimizedOuter 3527
BtnSelectedFont 3527
CaptionFontColor 3528
ControlColor 3528
ControlFrameColor 3528
ControlInactiveColor 3528
DisabledFontColor 3528
DocumentFontColor 3529
FontColor 3529
GroupFontColor 3529
HighlightColor 3529
InactiveTabFontColor 3530
QuickAccessToolbarMoreBorderColor 3530
QuickAccessToolbarMoreColor 3530
TRibbonLunaColorMap Members 3525
TRibbonLunaColorMap methods 3526
TRibbonLunaColorMap properties 3526
UpdateColors 3526
TRibbonLunaSkin class 3998
about TRibbonLunaSkin class 3998
GetLoadType 3999
GetLoadTypeValue 3999
TRibbonLunaSkin Members 3998
TRibbonLunaSkin methods 3999
TRibbonLunaStyleActionBars class 3999
about TRibbonLunaStyleActionBars class 3999
GetColorMapClass 4000
GetSkinClass 4001
GetStyleName 4001
TRibbonLunaStyleActionBars Members 4000
TRibbonLunaStyleActionBars methods 4000
TRibbonMainMenuItem class 3752
about TRibbonMainMenuItem class 3752
DrawShadowedText 3753
DrawText 3753
TRibbonMainMenuItem Members 3752
TRibbonMainMenuItem methods 3753
TRibbonMainMenuSeparator class 3754
about TRibbonMainMenuSeparator class 3754
CalcBounds 3755
TRibbonMainMenuSeparator Members 3754
TRibbonMainMenuSeparator methods 3755
TRibbonMenuItem class 3755
about TRibbonMenuItem class 3755
CalcBounds 3757
CalcLayout 3757
Click 3757
Create 3756
DrawBackground 3758
DrawBanner 3760
DrawGlyph 3758
DrawSelectedBackground 3758
DrawSeparator 3759
DrawShadowedText 3759
DrawText 3759
GetShowShortCut 3759
RibbonStyle 3760
TRibbonMenuItem Members 3755
TRibbonMenuItem methods 3756
TRibbonMenuItem properties 3760
TRibbonMenuSeparatorItem class 3760
about TRibbonMenuSeparatorItem class 3760

CalcBounds 3761
DrawSeparator 3762
TRibbonMenuSeparatorItem Members 3761
TRibbonMenuSeparatorItem methods 3761
TRibbonMenuSmallButtonControl class 3762
about TRibbonMenuSmallButtonControl class 3762
Create 3763
MouseUp 3764
TRibbonMenuSmallButtonControl Members 3762
TRibbonMenuSmallButtonControl methods 3764
TRibbonObsidianColorMap class 3530
about TRibbonObsidianColorMap class 3530
ActiveTabFontColor 3532
BevelMinimizedInner 3533
BevelMinimizedOuter 3533
BtnSelectedFont 3533
CaptionFontColor 3533
ControlColor 3533
ControlFrameColor 3534
ControlInactiveColor 3534
DisabledFontColor 3534
DocumentFontColor 3534
FontColor 3535
GroupFontColor 3535
HighlightColor 3535
InactiveTabFontColor 3535
QuickAccessToolbarMoreBorderColor 3536
QuickAccessToolbarMoreColor 3536
TRibbonObsidianColorMap Members 3531
TRibbonObsidianColorMap methods 3531
TRibbonObsidianColorMap properties 3532
UpdateColors 3532
TRibbonObsidianSkin class 4004
about TRibbonObsidianSkin class 4004
GetLoadType 4005
GetLoadTypeValue 4005
TRibbonObsidianSkin Members 4004
TRibbonObsidianSkin methods 4005
TRibbonObsidianStyleActionBars class 4005
about TRibbonObsidianStyleActionBars class 4005
GetColorMapClass 4006
GetSkinClass 4007
GetStyleName 4007
TRibbonObsidianStyleActionBars Members 4006
TRibbonObsidianStyleActionBars methods 4006
TRibbonOptionButtonControl class 3765
about TRibbonOptionButtonControl class 3765
DrawBackground 3766
TRibbonOptionButtonControl Members 3765
TRibbonOptionButtonControl methods 3766
TRibbonPage class 3536
about TRibbonPage class 3536
BiDiMode 3538
Caption 3539
Height 3541
Index 3542
KeyTip 3543
Name 3543
ParentBiDiMode 3545
TRibbonPage Members 3537
TRibbonPage properties 3538
TRibbonPopupGalleryBar class 3992
about TRibbonPopupGalleryBar class 3992
Assign 3994
CreateParams 3994
DisplayGallery 3995
TRibbonPopupGalleryBar Members 3993
TRibbonPopupGalleryBar methods 3993
TRibbonQATCustomizeMenu class 3968
about TRibbonQATCustomizeMenu class 3968
DisplayShadow 3969
DoAddCustomizeItems 3969
GetControlClass 3969
HideShadow 3969
NCPaint 3970
PositionPopup 3970
TRibbonQATCustomizeMenu Members 3968
TRibbonQATCustomizeMenu methods 3968

VisibleChanging 3970
TRibbonQATScrollButton class 3767
about TRibbonQATScrollButton class 3767
CalcBounds 3768
Click 3768
CMMouseEnter 3769
CMMouseLeave 3769
CursorInBtn 3771
DrawArrows 3769
DrawBackground 3769
DrawDown 3771
MouseMove 3770
RibbonStyle 3770
ScrollMode 3771
TRibbonQATScrollButton Members 3767
TRibbonQATScrollButton methods 3767
TRibbonQATScrollButton properties 3770
TRibbonQuickAccessPopupToolbar class 3545
about TRibbonQuickAccessPopupToolbar class 3545
TRibbonQuickAccessToolbar class 3546
about TRibbonQuickAccessToolbar class 3546
ActionManager 3548
OnMoreCommands 3548
TRibbonQuickAccessToolbar Members 3546
TRibbonQuickAccessToolbar properties 3548
VertMargin 3548
TRibbonRadioButtonControl class 3771
about TRibbonRadioButtonControl class 3771
CalcBounds 3773
DrawBackground 3773
DrawGlyph 3773
TRibbonRadioButtonControl Members 3772
TRibbonRadioButtonControl methods 3773
TRibbonRecentButtonControl class 3774
about TRibbonRecentButtonControl class 3774
AllowHotKeys 3775
TRibbonRecentButtonControl Members 3774
TRibbonRecentButtonControl methods 3775
TRibbonRecentDocuments class 3970
about TRibbonRecentDocuments class 3970
Caption 3972
Create 3971
Paint 3972
TRibbonRecentDocuments Members 3971
TRibbonRecentDocuments methods 3972
TRibbonRecentDocuments properties 3972
TRibbonRichContentItem class 3776
about TRibbonRichContentItem class 3776
CalcBounds 3777
CanCalculateHeight 3778
Create 3777
DrawBackground 3778
DrawGlyph 3778
DrawShadowedText 3779
DrawText 3779
GetItemHeight 3779
GetItemWidth 3780
GetMenuProps 3780
TRibbonRichContentItem Members 3776
TRibbonRichContentItem methods 3777
TRibbonRichContentMainMenuItem class 3780
about TRibbonRichContentMainMenuItem class 3780
CalcBounds 3782
CalcLayout 3782
Create 3781
DrawGlyph 3782
DrawText 3783
GetImageSize 3783
GetShowShortCut 3783
TRibbonRichContentMainMenuItem Members 3780
TRibbonRichContentMainMenuItem methods 3782
TRibbonSeparator class 3784
about TRibbonSeparator class 3784
CalcBounds 3785
Create 3784
DrawBackground 3785
DrawGlyph 3786
DrawShadowedText 3786

DrawText 3786
RibbonStyle 3787
TRibbonSeparator Members 3784
TRibbonSeparator methods 3785
TRibbonSilverColorMap class 3549
about TRibbonSilverColorMap class 3549
ActiveTabFontColor 3551
BevelMinimizedInner 3551
BevelMinimizedOuter 3551
BtnSelectedFont 3551
CaptionFontColor 3552
ControlColor 3552
ControlFrameColor 3552
ControlInactiveColor 3552
DisabledFontColor 3553
DocumentFontColor 3553
FontColor 3553
GroupFontColor 3553
HighlightColor 3554
InactiveTabFontColor 3554
QuickAccessToolbarMoreBorderColor 3554
QuickAccessToolbarMoreColor 3554
TRibbonSilverColorMap Members 3549
TRibbonSilverColorMap methods 3550
TRibbonSilverColorMap properties 3550
UpdateColors 3550
TRibbonSilverSkin class 4010
about TRibbonSilverSkin class 4010
GetLoadType 4011
GetLoadTypeValue 4011
TRibbonSilverSkin Members 4010
TRibbonSilverSkin methods 4011
TRibbonSilverStyleActionBars class 4011
about TRibbonSilverStyleActionBars class 4011
GetColorMapClass 4012
GetSkinClass 4013
GetStyleName 4013
TRibbonSilverStyleActionBars Members 4012
TRibbonSilverStyleActionBars methods 4012
TRibbonSmallButtonControl class 3787
about TRibbonSmallButtonControl class 3787
CalcBounds 3788
TRibbonSmallButtonControl Members 3787
TRibbonSmallButtonControl methods 3788
TRibbonSmallSplitButton class 3789
about TRibbonSmallSplitButton class 3789
CalcBounds 3792
CalcLayout 3792
Click 3793
CMMouseLeave 3793
Create 3790
CursorInSplit 3796
DrawBackground 3793
FCursorInSplit 3791
FSplitRect 3791
GetArrowPosition 3794
InSplit 3794
KeyTipClick 3794
MouseDown 3794
MouseMove 3795
TRibbonSmallSplitButton fields 3791
TRibbonSmallSplitButton Members 3789
TRibbonSmallSplitButton methods 3792
TRibbonSmallSplitButton properties 3795
UpdateSplitBounds 3795
TRibbonSpinButton class 3796
about TRibbonSpinButton class 3796
Align 3801
Anchors 3804
Constraints 3805
Control 3798
Create 3797
Ctl3D 3806
DownGlyph 3806
DownNumGlyphs 3806
DragCursor 3807
DragKind 3807
DragMode 3807

Enabled 3809
FocusControl 3809
KeyDown 3798
Loaded 3799
Notification 3799
OnDownClick 3820
OnDragDrop 3820
OnDragOver 3823
OnEndDock 3827
OnEndDrag 3827
OnEnter 3828
OnExit 3831
OnStartDock 3835
OnStartDrag 3835
OnUpClick 3836
ParentCtl3D 3810
ParentShowHint 3810
PopupMenu 3810
SetBounds 3800
ShowHint 3815
TabOrder 3817
TabStop 3818
TRibbonSpinButton events 3820
TRibbonSpinButton Members 3796
TRibbonSpinButton methods 3798
TRibbonSpinButton properties 3800
UpGlyph 3818
UpNumGlyphs 3818
Visible 3818
TRibbonSpinEdit class 3836
about TRibbonSpinEdit class 3836
Anchors 3845
AutoSelect 3846
AutoSize 3846
Button 3847
CMMouseenter 3839
CMMouseleave 3840
Color 3847
Constraints 3848
Create 3838
CreateParams 3840
CreateWnd 3840
Ctl3D 3849
Destroy 3838
DownClick 3841
DragCursor 3849
DragMode 3850
EditorEnabled 3851
Enabled 3851
Font 3852
GetChildren 3841
Increment 3853
IsValidChar 3841
KeyDown 3842
KeyPress 3842
MaxLength 3853
MaxValue 3854
MinValue 3854
OnChange 3866
OnClick 3867
OnDblClick 3868
OnDragDrop 3869
OnDragOver 3872
OnEndDrag 3875
OnEnter 3876
OnExit 3878
OnKeyDown 3882
OnKeyPress 3886
OnKeyUp 3887
OnMouseDown 3889
OnMouseMove 3891
OnMouseUp 3893
OnStartDrag 3896
ParentColor 3854
ParentCtl3D 3855
ParentFont 3855
ParentShowHint 3855
PopupMenu 3856

ReadOnly 3861
SetEnabled 3843
SetParent 3843
ShowHint 3861
TabOrder 3863
TabStop 3864
TRibbonSpinEdit events 3866
TRibbonSpinEdit Members 3836
TRibbonSpinEdit methods 3839
TRibbonSpinEdit properties 3844
UpClick 3844
Value 3864
Visible 3864
WMKillFocus 3844
TRibbonSplitMenuItem class 3896
about TRibbonSplitMenuItem class 3896
CalcBounds 3899
Click 3899
Create 3898
DrawBackground 3900
DrawSelectedBackground 3900
FMouseInSplit 3898
FSplitRect 3899
IsInSplit 3900
MouseInSplit 3902
MouseMove 3901
TRibbonSplitMenuItem fields 3898
TRibbonSplitMenuItem Members 3897
TRibbonSplitMenuItem methods 3899
TRibbonSplitMenuItem properties 3901
UpdateSplitBounds 3901
TRibbonStyleActionBars class 4022
about TRibbonStyleActionBars class 4022
Create 4023
Destroy 4023
DrawElement 4025
DrawElement method 4024
GetAddRemoveItemClass 4026
GetColorMapClass 4026
GetControlClass 4027
GetPopupClass 4027
GetScrollBtnClass 4027
GetSkinClass 4028
RibbonSkin 4028
TRibbonStyleActionBars Members 4023
TRibbonStyleActionBars methods 4024
TRibbonStyleActionBars properties 4028
TRibbonStyleMainPopupMenu class 3975
about TRibbonStyleMainPopupMenu class 3975
Create 3976
NCPaint 3976
PositionPopup 3976
TRibbonStyleMainPopupMenu Members 3975
TRibbonStyleMainPopupMenu methods 3976
TRibbonStyleMenuButton class 3902
about TRibbonStyleMenuButton class 3902
DrawBackground 3903
RibbonStyle 3903
TRibbonStyleMenuButton Members 3902
TRibbonStyleMenuButton methods 3902
TRibbonStylePopupMenu class 3977
about TRibbonStylePopupMenu class 3977
AlignControls 3978
CanAutoSize 3979
Create 3977
CreateControl 3979
Destroy 3978
GalleryResize 3980
TRibbonStylePopupMenu Members 3977
TRibbonStylePopupMenu methods 3978
TRibbonStylePopupMenu properties 3980
WMExitsizemove 3980
WMNCHitTest 3980
TRibbonTabItem class 3555
about TRibbonTabItem class 3555
Assign 3556
BestTabWidth 3558
Caption 3558

Destroy 3555
GetDisplayName 3557
KeyTip 3558
KeyTipForm 3558
MinTabwidth 3558
MinTruncatedWidth 3559
Modified 3559
Page 3559
TRibbonTabItem Members 3555
TRibbonTabItem methods 3556
TRibbonTabItem properties 3557
X 3559
Y 3560
TRibbonTabs class 3560
about TRibbonTabs class 3560
Add 3561
Create 3561
DesignerAdd 3564
FindGroup 3562
GetPageFromPageCaption 3562
IndexOfCaption 3562
Items 3564
Notify 3562
Ribbon 3565
SelectNext 3563
SelectPrior 3563
TabExists 3564
TRibbonTabs Members 3560
TRibbonTabs methods 3561
TRibbonTabs properties 3564
TRibbonTextControl class 3903
about TRibbonTextControl class 3903
CalcBounds 3904
DrawBackground 3904
DrawText 3905
TRibbonTextControl Members 3903
TRibbonTextControl methods 3904
TRibbonToolScrollButton class 3905
about TRibbonToolScrollButton class 3905
DrawArrows 3906
DrawBackground 3906
TRibbonToolScrollButton Members 3905
TRibbonToolScrollButton methods 3906
TRIO class 4056
about TRIO class 4056
AfterConstruction 4060
BeforeDestruction 4060
Converter 4063
Create 4058
Destroy 4058
DoAfterExecute 4061
DoBeforeExecute 4061
FConverter 4059
FIID 4059
FWebNode 4059
GetResponseStream 4062
IntfMD 4059
NewInstance 4062
OnAfterExecute 4065
OnBeforeExecute 4066
OnGetAttachment 4067
OnSendAttachment 4067
QueryInterface 4063
RefCount 4064
SOAPHeaders 4064
TRIO events 4065
TRIO fields 4058
TRIO Members 4057
TRIO methods 4060
TRIO properties 4063
WebNode 4065
TRowPosition
TCustomRibbonGroup class 3424
TRRollback
IIBSQLMonitorHook interface 573
TRRollbackRetaining
IIBSQLMonitorHook interface 573
TRStart

IIBSQLMonitorHook interface 573
Truncate
TIBBlobStream class 49
tsbGroupLeft enumeration member 3573
tsbGroupRight enumeration member 3573
tsbTabLeft enumeration member 3573
tsbTabRight enumeration member 3573
TSelectOptionsColumn class 2024
about TSelectOptionsColumn class 2024
Caption 2028
CaptionAttributes 2028
Custom 2028
DataSet 2029
DisplayRows 2029
FieldName 2030
IsColumn 2027
Items 2030
ItemsField 2031
Style 2031
StyleRule 2032
TabIndex 2032
TSelectOptionsColumn Members 2025
TSelectOptionsColumn methods 2027
TSelectOptionsColumn properties 2027
Values 2032
ValuesField 2033
TServerCollection class 2466
about TServerCollection class 2466
Create 2467
FindServer 2468
GetBalancedName 2468
GetNextName 2469
Items 2470
TServerCollection Members 2467
TServerCollection methods 2468
TServerCollection properties 2469
TServerEventDispatch class 2720
_AddRef 2721
_Release 2721
about TServerEventDispatch class 2720
Create 2720
GetIDsOfNames 2721
GetTypeInfo 2722
GetTypeInfoCount 2722
Invoke 2723
QueryInterface 2723
Server 2724
ServerDisconnect 2723
TServerEventDispatch Members 2720
TServerEventDispatch methods 2721
TServerEventDispatch properties 2724
TServerItem class 2470
about TServerItem class 2470
ComputerName 2472
Create 2471
Enabled 2473
GetDisplayName 2472
HasFailed 2474
Port 2474
TServerItem Members 2471
TServerItem methods 2472
TServerItem properties 2472
TSharedConnection class 1309
about TSharedConnection class 1309
AfterConnect 1315
AfterDisconnect 1315
BeforeConnect 1315
BeforeDisconnect 1316
ChildName 1314
ConnectEvent 1311
Create 1310
Destroy 1311
DoConnect 1311
DoDisconnect 1312
GetConnected 1312
GetServer 1313
Notification 1313
ParentConnection 1314

TSharedConnection events 1314
TSharedConnection Members 1310
TSharedConnection methods 1311
TSharedConnection properties 1313
TSimpleObjectBroker class 2475
about TSimpleObjectBroker class 2475
Create 2476
Destroy 2476
GetComputerForGUID 2477
GetComputerForProgID 2477
GetPortForComputer 2478
LoadBalanced 2480
LoadFromStream 2479
SaveToStream 2479
ServerData 2481
Servers 2481
SetConnectStatus 2479
TSimpleObjectBroker Members 2475
TSimpleObjectBroker methods 2476
TSimpleObjectBroker properties 2480
TSOAPAttachment class 895
about TSOAPAttachment class 895
CacheFile 904
CacheFilePersist 905
ContentType 905
Create 897
Destroy 898
Encoding 906
GetSourceStream 899
Headers 906
Init 899
InternalSetCacheFile 900
InternalSetSourceStream 900
ObjectToSOAP 900
Ownership 907
SaveToFile 901
SaveToStream 901
SetOwnership 902
SetSourceFile 902
SetSourceStream 903
SetSourceString 903
SOAPToObject 903
SourceStream 907
SourceString 908
TSOAPAttachment Members 897
TSOAPAttachment methods 898
TSOAPAttachment properties 904
TSOAPDomConv class 2758
about TSOAPDomConv class 2758
AddAttachment 2763
AddMultiRefNode 2763
AddNamespaceDecl 2763
AddObjectAsWriting 2763
CheckEncodingStyle 2764
ChildNodesAreNull 2764
ConvertByteArrayToSoap 2764
ConvertEnumToSoap 2764
ConvertNativeArrayToSoap 2765
ConvertNativeDataToSoap 2765
ConvertObjectToSOAP 2765
ConvertSoapToEnum 2766
ConvertSoapToNativeArray 2766
ConvertSoapToNativeArrayElem 2766
ConvertSoapToNativeData 2766
ConvertSOAPToObject 2767
ConvertSoapToVariant 2767
ConvertVariantToSoap 2768
Create 2760
CreateMultiRefNode 2768
CreateNULLNode 2768
CreateObjectNode 2768
CreateScalarNodeXS 2769
CreateTypedNode 2769
Destroy 2761
FinalizeMultiRefNodes 2769
FindAttachment 2769
FindMultiRefNodeByID 2770
FindMultiRefNodeByInstance 2770

FindNodeByHREF 2770
FindPrefixForURI 2770
GetDataNode 2771
GetElementType 2771
GetNewID 2771
GetNodeAsText 2771
GetObjectPropAsText 2771
GetOptions 2772
GetTypeBySchemaNS 2772
InitObjectFromSOAP 2772
IsNodeAVarArray 2773
IsObjectWriting 2773
LoadObject 2773
MakeArrayNode 2774
MakeArrayNode method 2773
MultiRefObject 2774
NodeIsNULL 2774
ObjectMemberNoShow 2775
ObjInstanceToSOAP 2775
OnMemberDataNotReceived 2782
OnUnhandledNode 2782
Options 2781
ReadHeader 2775
ReadRectDynArray 2776
ReadRectDynArrayElem 2776
ReadRow 2776
ReadVarArrayDim 2776
ReadVariant 2777
RemoveObjectAsWriting 2777
ResetMultiRef 2777
SerializationOptions 2778
SerializationOptions method 2777
SetNodeType 2778
SetObjectPropFromText 2778
SetOptions 2779
TSOAPDomConv events 2782
TSOAPDomConv Members 2759
TSOAPDomConv methods 2761
TSOAPDomConv properties 2781
UnhandledNode 2779
WriteHeader 2779
WriteNonRectDynArray 2779
WriteNonRectDynArrayElem 2780
WriteRectDynArray 2780
WriteRectDynArrayElem 2780
WriteVarArray 2780
WriteVarArrayAsB64 2781
WriteVariant 2781
TSOAPHeader class 908
about TSOAPHeader class 908
Actor 911
MustUnderstand 912
ObjectToSOAP 910
SOAPToObject 911
TSOAPHeader Members 909
TSOAPHeader methods 910
TSOAPHeader properties 911
TSOAPHeaders class 912
about TSOAPHeaders class 912
Get 914
Get method 914
GetOwnsSentHeaders 914
Send 915
SendAt 915
SendCount 916
SetOwnsSentHeaders 916
TSOAPHeaders Members 913
TSOAPHeaders methods 913
TSOAPHeadersBase class 916
about TSOAPHeadersBase class 916
FHeadersInbound 917
FHeadersOutBound 918
SetHeadersInOut 918
TSOAPHeadersBase fields 917
TSOAPHeadersBase Members 917
TSOAPHeadersBase methods 918
TSQLResolver class 3196
about TSQLResolver class 3196

Create 3198
Destroy 3198
DoDelete 3199
DoExecSQL 3200
DoGetValues 3200
DoInsert 3200
DoUpdate 3201
FreeTreeData 3201
GenDeleteSQL 3201
GenInsertSQL 3202
GenSelectSQL 3202
GenUpdateSQL 3202
GenWhereSQL 3202
GetProvider 3203
InitializeConflictBuffer 3203
InitTreeData 3203
InternalDoUpdate 3204
Provider 3204
TSQLResolver Members 3197
TSQLResolver methods 3199
TSQLResolver properties 3204
UseFieldInUpdate 3204
UseFieldInWhere 3204
tssDark enumeration member 3574
tssHidden enumeration member 3574
tssLight enumeration member 3574
tssMedium enumeration member 3574
TStaticListAction class 1054
about TStaticListAction class 1054
Active 1056
Caption 1057
Enabled 1057
HelpContext 1058
Hint 1058
Images 1058
ItemIndex 1059
Items 1059
OnGetItem 1061
OnHint 1061
OnItemSelected 1062
OnUpdate 1062
SecondaryShortCuts 1060
ShortCut 1060
TStaticListAction events 1061
TStaticListAction Members 1055
TStaticListAction properties 1056
Visible 1060
TStaticListItems class 1063
about TStaticListItems class 1063
Notify 1065
TStaticListItems Members 1064
TStaticListItems methods 1064
Update 1065
TStatusColumn class 2033
about TStatusColumn class 2033
Caption 2036
CaptionAttributes 2036
Custom 2037
IsColumn 2035
Style 2037
StyleRule 2038
TStatusColumn Members 2034
TStatusColumn methods 2035
TStatusColumn properties 2036
TStream class 3005
about TStream class 3005
CopyFrom 3006
GetPosition 3006
GetSize 3006
Position 3008
Read 3007
Seek 3007
SetPosition 3007
SetSize 3007
Size 3008
TStream Members 3005
TStream methods 3006
TStream properties 3008

Write 3007
TStringHash class 783
about TStringHash class 783
Add 784
Clear 784
Create 783
Destroy 784
Find 785
HashOf 785
Modify 785
Remove 785
TStringHash Members 783
TStringHash methods 784
ValueOf 786
TSubmitQueryButton class 2038
about TSubmitQueryButton class 2038
Create 2040
EventContent 2040
TSubmitQueryButton Members 2039
TSubmitQueryButton methods 2040
TTabChangeEvent
TCustomRibbon class 3400
TTextAreaColumn class 2041
about TTextAreaColumn class 2041
Caption 2044
CaptionAttributes 2044
Custom 2045
DisplayRows 2045
DisplayWidth 2046
FieldName 2046
IsColumn 2043
ReadOnly 2046
Style 2047
StyleRule 2047
TabIndex 2048
TTextAreaColumn Members 2041
TTextAreaColumn methods 2043
TTextAreaColumn properties 2044
Wrap 2048
TTextColumn class 2049
about TTextColumn class 2049
Caption 2052
CaptionAttributes 2052
Custom 2053
DisplayWidth 2053
FieldName 2053
IsColumn 2051
ReadOnly 2054
Style 2054
StyleRule 2055
TabIndex 2055
TTextColumn Members 2049
TTextColumn methods 2051
TTextColumn properties 2051
TTimerSpeedButton class 3906
about TTimerSpeedButton class 3906
ButtonStyle 3910
CMMouseenter 3908
CMMouseleave 3908
Create 3907
Destroy 3908
MouseDown 3909
MouseInControl 3910
MouseUp 3909
Paint 3910
TimeBtnState 3911
TTimerSpeedButton Members 3907
TTimerSpeedButton methods 3908
TTimerSpeedButton properties 3910
ttSystem enumeration member 626
ttView enumeration member 626
TUndoButton class 2055
about TUndoButton class 2055
Create 2057
TUndoButton Members 2056
TUpdateTree class 3205
about TUpdateTree class 3205
Clear 3207

Create 3206
Data 3209
Delta 3210
Destroy 3207
DetailCount 3210
DetailList 3210
Details 3210
DoUpdates 3207
ErrorDS 3211
HasErrors 3211
InitData 3208
InitDelta 3208
InitDelta method 3208
InitErrorPacket 3209
IsNested 3211
Name 3211
Opened 3211
Parent 3212
RefreshData 3209
Resolver 3212
Source 3212
TUpdateTree Members 3206
TUpdateTree methods 3207
TUpdateTree properties 3209
TUserInfo class 472
about TUserInfo class 472
FirstName 474
GroupID 474
LastName 474
MiddleName 474
TUserInfo fields 473
TUserInfo Members 473
UserID 474
UserName 475
TVersionInfo class 475
about TVersionInfo class 475
ServerImplementation 476
ServerVersion 476
ServiceVersion 476
TVersionInfo fields 476
TVersionInfo Members 475
TVirtualListAction class 1066
about TVirtualListAction class 1066
Active 1068
Caption 1068
Enabled 1068
HelpContext 1069
Hint 1069
Images 1070
ItemIndex 1070
OnGetItem 1072
OnGetItemCount 1072
OnHint 1073
OnItemSelected 1073
SecondaryShortCuts 1070
ShortCut 1071
TVirtualListAction events 1071
TVirtualListAction Members 1066
TVirtualListAction properties 1067
Visible 1071
TWebButton class 2058
about TWebButton class 2058
Caption 2062
Content 2060
Create 2059
Custom 2062
GetCaption 2060
GetLayoutAttributes 2060
ImplContent 2061
SetCaption 2061
Style 2063
StyleRule 2063
TWebButton Members 2059
TWebButton methods 2059
TWebButton properties 2062
TWebControlGroup class 2064
about TWebControlGroup class 2064
AddElements 2066

CanAddClass 2067
ChangeFieldControls 2067
Content 2068
Create 2065
DefaultWebComponents 2072
Destroy 2066
GetDefaultWebComponents 2068
GetLayoutAttributes 2069
GetSubComponents 2069
ImplAddElements 2069
ImplCanAddClass 2070
ImplContent 2070
ImplGetSubComponents 2071
SetDefaultWebComponents 2071
TWebControlGroup Members 2065
TWebControlGroup methods 2066
TWebControlGroup properties 2072
WebFieldControls 2072
TWebDataDisplay class 2073
about TWebDataDisplay class 2073
Caption 2083
CaptionAttributes 2084
CaptionPosition 2084
Content 2076
ControlContent 2077
Create 2075
Custom 2084
Destroy 2075
EventContent 2077
FormatCaption 2077
GetCaption 2078
GetHTMLControlName 2078
GetHTMLForm 2078
GetLayoutAttributes 2079
GetXmlDisplayName 2079
GetXmlRowSetName 2080
ImplContent 2080
ImplGetHTMLControlName 2081
IsColumn 2081
IsQueryField 2081
LabelContent 2082
LayoutAttributes 2085
SetCaption 2082
SetCaptionAttributes 2083
Style 2085
StyleRule 2086
TabIndex 2086
TWebDataDisplay Members 2074
TWebDataDisplay methods 2075
TWebDataDisplay properties 2083
TWebDataInput class 2087
about TWebDataInput class 2087
EventContent 2089
FieldName 2094
FindAssociatedField 2090
GetCaption 2090
GetFieldName 2090
GetParamName 2090
GetRowSetFieldAttributes 2091
ImplGetFieldName 2091
ImplGetHTMLControlName 2091
ImplGetRowSetFieldAttributes 2091
ImplRestoreDefaults 2092
ParamName 2094
RestoreDefaults 2092
RestoreFieldDefaults 2092
SetCaption 2092
SetFieldName 2093
SetParamName 2093
TWebDataInput Members 2088
TWebDataInput methods 2089
TWebDataInput properties 2093
ValidateField 2093
TWebDisplay class 2095
about TWebDisplay class 2095
Changed 2096
ChangedDisplayComponent 2096
Create 2095

DisplayComponent 2097
GetDisplayComponent 2096
Notification 2097
OnChange 2098
Parent 2097
SetDisplayComponent 2097
TWebDisplay events 2098
TWebDisplay Members 2095
TWebDisplay methods 2096
TWebDisplay properties 2097
TWebDisplayReferenceGroup class 2098
about TWebDisplayReferenceGroup class 2098
Create 2100
Destroy 2100
GetWebDisplayComponent 2100
Notification 2101
SetWebDisplayComponent 2101
TWebDisplayReferenceGroup Members 2099
TWebDisplayReferenceGroup methods 2100
TWebDisplayReferenceGroup properties 2101
WebDisplay 2102
WebDisplayComponent 2102
TWebForm class 2102
about TWebForm class 2102
AddElements 2105
CanAddClass 2105
Content 2106
Create 2104
Custom 2110
Destroy 2104
GetDefaultWebComponents 2106
GetHTMLFormName 2107
GetHTMLFormVarName 2107
GetLayoutAttributes 2107
GetSubComponents 2108
GetVisibleFields 2108
ImplAddElements 2108
ImplCanAddClass 2109
ImplContent 2109
SetDefaultWebComponents 2110
Style 2111
StyleRule 2111
TWebForm Members 2103
TWebForm methods 2104
TWebForm properties 2110
VisibleFields 2112
WebComponents 2112
TWebListInput class 2113
about TWebListInput class 2113
AddElements 2117
ControlContent 2117
Create 2116
DataSet 2119
Destroy 2116
FormatInputs 2117
GetSubComponents 2118
ImplAddElements 2118
Items 2120
ItemsField 2120
Notification 2118
SetDataSet 2119
SetItems 2119
SetValues 2119
TWebListInput Members 2114
TWebListInput methods 2116
TWebListInput properties 2119
Values 2121
ValuesField 2121
TWebRadioGroupInput class 2122
about TWebRadioGroupInput class 2122
AddAttributes 2125
Create 2125
DisplayColumns 2127
DisplayWidth 2127
EventContent 2126
FormatInputs 2126
GetCheckIndex 2126
ImplGetRowSetFieldAttributes 2126

ReadOnly 2128
RestoreFieldDefaults 2126
TWebRadioGroupInput Members 2123
TWebRadioGroupInput methods 2125
TWebRadioGroupInput properties 2127
TWebRequestFile class 3332
about TWebRequestFile class 3332
Create 3333
Destroy 3334
GetContentType 3334
GetFieldName 3334
GetFileName 3335
GetStream 3335
TWebRequestFile Members 3333
TWebRequestFile methods 3334
TWebRequestFiles class 3336
about TWebRequestFiles class 3336
Add 3338
Add method 3337
Create 3336
Destroy 3337
GetCount 3338
GetItem 3339
TWebRequestFiles Members 3336
TWebRequestFiles methods 3337
TWebRequestFileStream class 3339
about TWebRequestFileStream class 3339
Create 3340
TWebRequestFileStream Members 3339
TWebRequestFileStream methods 3340
Write 3340
TWebSelectOptionsInput class 2128
about TWebSelectOptionsInput class 2128
AddAttributes 2131
Create 2131
DisplayRows 2132
EventContent 2132
FormatInputs 2132
GetSelectIndex 2132
TWebSelectOptionsInput Members 2129
TWebSelectOptionsInput methods 2131
TWebSelectOptionsInput properties 2132
TWebStatus class 2133
about TWebStatus class 2133
Caption 2137
CaptionAttributes 2137
CaptionPosition 2138
ControlContent 2135
Create 2135
DisplayWidth 2138
GetCaption 2136
Identifier 2136
ImplGetHTMLControlName 2136
TWebStatus Members 2134
TWebStatus methods 2135
TWebStatus properties 2137
TWebTextAreaInput class 2138
about TWebTextAreaInput class 2138
AddAttributes 2142
ControlContent 2142
Create 2141
DisplayRows 2143
DisplayWidth 2144
EventContent 2142
ImplGetRowSetFieldAttributes 2143
ReadOnly 2144
RestoreFieldDefaults 2143
TWebTextAreaInput Members 2140
TWebTextAreaInput methods 2141
TWebTextAreaInput properties 2143
Wrap 2144
TWebTextInput class 2145
about TWebTextInput class 2145
AddAttributes 2148
ControlContent 2148
Create 2147
Custom 2149
DisplayWidth 2150

ImplGetRowSetFieldAttributes 2149
ReadOnly 2150
RestoreFieldDefaults 2149
Style 2150
StyleRule 2151
TWebTextInput Members 2146
TWebTextInput methods 2148
TWebTextInput properties 2149
TXMLButton class 2151
about TXMLButton class 2151
Create 2153
DefaultCaption 2156
Destroy 2153
GetCaption 2154
GetXMLBroker 2154
GetXMLUseParent 2154
Notification 2154
SetXMLBroker 2155
SetXMLUseParent 2155
TXMLButton Members 2152
TXMLButton methods 2153
TXMLButton properties 2155
XMLBroker 2156
XMLData 2156
XMLUseParent 2156
TXMLData class 2157
about TXMLData class 2157
Changed 2159
ChangedXMLBroker 2159
Create 2158
GetXMLBroker 2159
Notification 2159
OnChange 2161
Parent 2160
SetXMLBroker 2160
TXMLData events 2161
TXMLData Members 2157
TXMLData methods 2158
TXMLData properties 2160
XMLBroker 2160
TXMLDataParent class 2161
about TXMLDataParent class 2161
ChangedUseParent 2163
ChangedXMLBroker 2163
GetXMLBroker 2163
ParentXMLBroker 2163
SetUseParent 2163
TXMLDataParent Members 2162
TXMLDataParent methods 2162
TXMLDataParent properties 2164
UseParent 2164
TXMLDataSet class 2164
_AddRef 2167
_Release 2167
about TXMLDataSet class 2164
ChangedDataSetField 2167
ChangedXMLBroker 2167
CompareXMLData 2167
ConnectionChange 2168
Create 2165
CreateDataSet 2168
DataSet 2170
DataSetField 2171
DataSetPath 2171
Destroy 2166
FindDataSet 2168
GetDataSet 2168
GetDataSetField 2169
GetDataSetPath 2169
Parent 2171
QueryInterface 2169
SetDataSetField 2169
SetXMLBroker 2169
TXMLDataSet Members 2165
TXMLDataSet methods 2166
TXMLDataSet properties 2170
WrapperDataChange 2170
TXMLDataSetParent class 2172

about TXMLDataSetParent class 2172
ChangedDataSetField 2174
ChangedUseParent 2174
ChangedXMLBroker 2174
GetDataSetField 2174
GetXMLBroker 2174
ParentDataSetField 2175
ParentXMLBroker 2175
ParentXMLDataSet 2175
SetUseParent 2175
TXMLDataSetParent Members 2172
TXMLDataSetParent methods 2173
TXMLDataSetParent properties 2175
UseParent 2176
TXMLDisplayGroup class 2176
about TXMLDisplayGroup class 2176
Create 2178
CreateDefaultFields 2179
Destroy 2178
EnableValidateFields 2186
FindField 2180
GetDataSet 2180
GetDataSetField 2180
GetEnableValidateFields 2180
GetFieldsList 2181
GetIsMultipleRecordView 2181
GetVisibleFields 2181
GetXMLBroker 2181
GetXMLDataSet 2182
GetXMLDisplayName 2182
GetXMLRowSetName 2182
HasStatusField 2182
ImplAddElements 2183
ImplGetSubComponents 2183
ImplIsMultipleRecordView 2183
IsFieldInUse 2184
Notification 2184
SetDataSetField 2184
SetEnableValidateFields 2185
SetXMLBroker 2185
TXMLDisplayGroup Members 2177
TXMLDisplayGroup methods 2179
TXMLDisplayGroup properties 2186
ValidateDataSet 2185
ValidateFields 2185
VisibleFields 2186
XMLBroker 2187
XMLDataChange 2185
XMLDataSet 2187
XMLDataSetField 2188
TXMLDisplayParent class 2188
about TXMLDisplayParent class 2188
ChangedDisplayComponent 2190
ChangedUseParent 2190
Create 2189
GetDisplayComponent 2190
SetUseParent 2190
TXMLDisplayParent Members 2189
TXMLDisplayParent methods 2189
TXMLDisplayParent properties 2190
UseParent 2191
TXMLDisplayReferenceButton class 2191
about TXMLDisplayReferenceButton class 2191
Create 2193
Destroy 2193
GetDisplayComponent 2194
GetDisplayComponentParent 2194
GetXmlDisplayName 2194
Notification 2194
SetDisplayComponent 2195
SetDisplayComponentParent 2195
TXMLDisplayReferenceButton Members 2192
TXMLDisplayReferenceButton methods 2193
TXMLDisplayReferenceButton properties 2195
XMLComponent 2195
XMLDisplay 2196
XMLUseParent 2196
TXMLDisplayReferenceGroup class 2197

about TXMLDisplayReferenceGroup class 2197
ImplAddElements 2198
TXMLDisplayReferenceGroup Members 2197
TXMLDisplayReferenceGroup methods 2198
TXMLDisplayReferenceGroup properties 2199
XMLComponent 2199
TypeInfoToXSD
TRemotableTypeRegistry class 887

U

uaAbort enumeration member 151
uaApplied enumeration member 151
uaApply enumeration member 151
uaFail enumeration member 151
uaRetry enumeration member 151
uaSkip enumeration member 151
UM_CHECKSIZE
TCustomRibbon class 3367
UM_DISPLAYKEYTIPS
TCustomRibbon class 3368
UMCheckResize
TCustomRibbon class 3391
UMDisplayKeyTips
TCustomRibbon class 3391
UMFadeControl
TRibbonApplicationMenuButton class 3611
Undelete
TIBCustomDataSet class 92
UnhandledNode
TSOAPDomConv class 2779
UniDirectional
TIBCustomDataSet class 107
TIBDataSet class 132
TIBQuery class 393
TIBTable class 622
UnInstallCheck
TIBUnInstall class 324
UnInstallExecute
TIBUnInstall class 324
UnInstallFile
TIBInstall class 316
TIBUnInstall class 324
UniqueRelationName
TIBSQL class 535
TIBXSQLDA class 544
UnknownAdvise enumeration member 483
UnknownState enumeration member 484
UnLoadKey
TRegistry class 3280
UnLock
TRemotableTypeRegistry class 888
Unmerge
TMainMenu class 1326
UnPrepare
TIBDataSet class 121
TIBQuery class 386
TIBStoredProc class 592
UnRegisterChanges
TCustomImageList class 674
UnRegisterEvents
IIBEventNotifier interface 222
TIBEvents class 262
UnRegisterInterface
TInvokableClassRegistry class 867
UnRegisterInvokableClass
TInvokableClassRegistry class 868
UnregisterMonitor
IIBSQLMonitorHook interface 573
UnRegisterProvider
IProviderContainer interface 3214
UnRegisterXSClass
TRemotableTypeRegistry class 888
UnRegisterXSInfo
TRemotableTypeRegistry class 889
UpClick
TRibbonSpinEdit class 3844
Update
TListControlItems class 1051

TStaticListItems class 1065
UpdateActionManager
TCustomRibbonPage class 3436
UpdateColors
TRibbonLunaColorMap class 3526
TRibbonObsidianColorMap class 3532
TRibbonSilverColorMap class 3550
UpdateCount
TIBDatabaseInfo class 254
Updated
TMediaPlayer class 2331
UpdateFile
TCustomIniFile class 736
TIniFile class 762
TMemIniFile class 778
TRegistryIniFile class 3317
UpdateIndexDefs
TIBTable class 614
UpdateItems
TMenu class 1343
TMenuItem class 1392
UpdateKind
TPacketDataSet class 3194
UpdateMode
TBaseProvider class 3089
TDataSetProvider class 3163
TIBCustomDataSet class 107
UpdateObject
TIBCustomDataSet class 107
TIBDataSet class 132
TIBQuery class 393
TIBTable class 623
TOleContainer class 2551
UpdateQATPosition
TCustomRibbon class 3391
UpdateQATWidth
TCustomRibbonQuickAccessToolbar class 3451
UpdateRecord
TBaseProvider class 3087
TDataSetProvider class 3158
UpdateRecordTypes
TIBCustomDataSet class 108
UpdateRegistry
TAutomation class 2506
TRegistryClass class 2512
UpdateScroll
TPanningWindow class 698
UpdateScrollButtons
TCustomRibbonPage class 3436
UpdatesPending
TIBCustomDataSet class 108
UpdateSplitBounds
IRibbonSplitButton interface 3923
TRibbonLargeSplitButton class 3751
TRibbonSmallSplitButton class 3795
TRibbonSplitMenuItem class 3901
UpdateStatus
TIBCustomDataSet class 92
UpdateSystemMenuInfo
TCustomRibbon class 3391
UpdateTree
TCustomResolver class 3135
UpdateVerbs
TOleContainer class 2551
UpGlyph
TRibbonSpinButton class 3818
UpNumGlyphs
TRibbonSpinButton class 3818
URIToClass
TRemotableTypeRegistry class 889
URIToInfo
TRemotableTypeRegistry class 890
UseAllSpace enumeration member 480
UseBroker
TMidasConnection class 1787
UseCurValues
TPacketDataSet class 3195
UseCustomFrame

TCustomRibbon class 3399
TRibbon class 3497
UseFieldInUpdate
TSQLResolver class 3204
UseFieldInWhere
TSQLResolver class 3204
UseParent
TXMLDataParent class 2164
TXMLDataSetParent class 2176
TXMLDisplayParent class 2191
UserID
TIBSecurityService class 450
TUserInfo class 474
UseRightToLeftAlignment
TPopupMenu class 1435
UserInfo
TIBSecurityService class 451
UserInfoCount
TIBSecurityService class 451
UserName
TIBSecurityService class 451
TUserInfo class 475
UserNames
TIBDatabaseInfo class 255
UStrToClear
TDataContext class 817

V

Validate
TCustomMaskEdit class 1091
TKeyTipValidator class 3601
ValidateDatabase
TIBEvents class 263
ValidateDataSet
IDataSetFields interface 2207
IQueryFields interface 2221
TXMLDisplayGroup class 2185
ValidateDB enumeration member 485
ValidateEdit
TCustomMaskEdit class 1091
ValidateError
TCustomMaskEdit class 1091
ValidateField
IValidateField interface 2224
TWebDataInput class 2093
ValidateFields
IValidateFields interface 2225
TXMLDisplayGroup class 2185
ValidateFull enumeration member 485
ValidOrientations
TCustomRibbonQuickAccessPopupToolbar class 3441
ValidTab
TCustomRibbon class 3392
VAlign
TCaptionAttributes class 1807
TGridRowAttributes class 1934
Value
ISharedProperty interface 2429
TIBXSQLVAR class 555
TRibbonSpinEdit class 3864
ValueExists
TCustomIniFile class 736
TRegistry class 3281
ValueName
TIBGeneratorField class 144
ValueOf
TStringHash class 786
Values
TFieldRadioGroup class 1883
TFieldSelectOptions class 1893
TQueryRadioGroup class 1988
TQuerySelectOptions class 1999
TSelectOptionsColumn class 2032
TWebListInput class 2121
ValuesField
TFieldRadioGroup class 1884
TFieldSelectOptions class 1893
TQueryRadioGroup class 1988

TQuerySelectOptions class 2000
TSelectOptionsColumn class 2033
TWebListInput class 2121
ValueToString
TEnumPropDesc class 2634
VariantToInfo
TRemotableTypeRegistry class 890
Vars
TIBXSQLDA class 544
VarToClear
TDataContext class 818
Verbose
TIBBackupRestoreService class 408
Version
TIBDatabaseInfo class 255
Version enumeration member 479
VersionInfo
TIBServerProperties class 457
VertMargin
TRibbonApplicationMenuBar class 3958
TRibbonQuickAccessToolbar class 3548
Visible
TMaskEdit class 1139
TMediaPlayer class 2363
TMenuItem class 1416
TOleContainer class 2581
TOutline class 2904
TRibbonComboBox class 3646
TRibbonSpinButton class 3818
TRibbonSpinEdit class 3864
TStaticListAction class 1060
TVirtualListAction class 1071
VisibleButtons
TCustomDataNavigator class 1822
TCustomQueryButtons class 1839
TMediaPlayer class 2365
VisibleChanging
TRibbonQATCustomizeMenu class 3970
VisibleFields
IDataSetFields interface 2208
TWebForm class 2112
TXMLDisplayGroup class 2186
VisibleMenu
TRibbonApplicationMenuBar class 3959
VisibleMenuCount
TRibbonApplicationMenuBar class 3959

W

Wait
TMediaPlayer class 2365
WarningsContent
TInetXPageElements class 2284
WebComponents
TWebForm class 2112
WebContent
TCustomInetXPageProducer class 2281
WebDisplay
TWebDisplayReferenceGroup class 2102
WebDisplayComponent
IWebDisplayReference interface 2228
TWebDisplayReferenceGroup class 2102
WebFieldControls
TWebControlGroup class 2072
WebNode
TRIO class 4065
WebPageItems
TBasePageItemsProducer class 2969
TInetXPageProducer class 2292
Width
TCustomImageList class 684
TRibbonApplicationMenuBar class 3959
Window
TPopupList class 1430
WindowHandle
TMenu class 1351
WMActivate
TRibbonGroupDockForm class 3520
WMEraseBkgnd

TCustomRibbon class 3392
WMExitsizemove
TRibbonStylePopupMenu class 3980
WMGetDlgCode
TCustomRibbon class 3392
WMKillFocus
TCustomRibbon class 3392
TCustomRibbonComboBox class 3590
TRibbonSpinEdit class 3844
WMLButtonDblClk
TRibbonApplicationMenuButton class 3611
WMNCHitTest
TRibbonStylePopupMenu class 3980
WMSetFocus
TCustomRibbonComboBox class 3590
WMSettingChange
TCustomRibbon class 3393
WMSysCommand
TRibbonApplicationMenuBar class 3949
WMWindowPosChanged
TCustomRibbon class 3393
WndProc
TOleControl class 2691
TPanningWindow class 699
TPopupList class 1429
wrAbort enumeration member 332
Wrap
TFieldTextArea class 1912
TQueryTextArea class 2021
TTextAreaColumn class 2048
TWebTextAreaInput class 2144
WrapperDataChange
TXMLDataSet class 2170
wrContinue enumeration member 332
Write
TFileStream class 2996
TIBBlobStream class 50
TIBDSBlobStream class 141
TResourceStream class 3005
TStream class 3007
TWebRequestFileStream class 3340
WriteBinaryData
TRegistry class 3281
WriteBinaryStream
TCustomIniFile class 737
TRegistryIniFile class 3317
WriteBool
TCustomIniFile class 737
TRegIniFile class 3244
TRegistry class 3282
WriteClient
TISAPIRequest class 975
WriteColumns
TIBBatchOutput class 499
TIBOutputDelimitedFile class 510
TIBOutputRawFile class 513
WriteCurrency
TRegistry class 3282
WriteData
TCustomImageList class 675
TJPEGImage class 1007
WriteDataSet
TDataPacketWriter class 3143
WriteDate
TCustomIniFile class 741
TRegistry class 3282
TRegistryIniFile class 3317
WriteDateTime
TCustomIniFile class 741
TRegistry class 3283
TRegistryIniFile class 3318
WriteExpandString
TRegistry class 3286
WriteFloat
TCustomIniFile class 742
TRegistry class 3288
TRegistryIniFile class 3319
WriteHeader

TSOAPDomConv class 2779
WriteHeaders
TISAPIRequest class 975
WriteInteger
TCustomIniFile class 742
TRegIniFile class 3245
TRegistry class 3289
TRegistryIniFile class 3319
WriteMetaData
TDataPacketWriter class 3143
WriteNode
TOutlineNode class 2950, 2951
WriteNonRectDynArray
TSOAPDomConv class 2779
WriteNonRectDynArrayElem
TSOAPDomConv class 2780
WriteRectDynArray
TSOAPDomConv class 2780
WriteRectDynArrayElem
TSOAPDomConv class 2780
Writes
TIBDatabaseInfo class 255
WriteString
TCustomIniFile class 745
TIniFile class 762
TISAPIRequest class 976
TMemIniFile class 779
TRegIniFile class 3245
TRegistry class 3289
TRegistryIniFile class 3322
WriteTime
TCustomIniFile class 748
TRegistry class 3292
TRegistryIniFile class 3325
WriteVarArray
TSOAPDomConv class 2780
WriteVarArrayAsB64
TSOAPDomConv class 2781
WriteVariant
TSOAPDomConv class 2781
WritingCurValues
TPacketDataSet class 3196
wrOff enumeration member 2239
wrPhysical enumeration member 2239
wrVirtual enumeration member 2239
WSDLView
TOPToSoapDomConvert class 2758
WStrToClear
TDataContext class 818

X

X
TRibbonTabItem class 3559
XMLBroker
IDataSetComponent interface 2203
TApplyUpdatesButton class 1803
TXMLButton class 2156
TXMLData class 2160
TXMLDisplayGroup class 2187
XMLComponent
TDataNavigator class 1859
TDataSetButton class 1866
TQueryButton class 1961
TXMLDisplayReferenceButton class 2195
TXMLDisplayReferenceGroup class 2199
XMLData
TXMLButton class 2156
XMLDataChange
TXMLDisplayGroup class 2185
XMLDataSet
IDataSetComponent interface 2203
TXMLDisplayGroup class 2187
XMLDataSetField
TXMLDisplayGroup class 2188
XMLDisplay
TXMLDisplayReferenceButton class 2196
XMLDisplayName
IXMLDisplay interface 2231

XMLMethodName
TDataSetButton class 1862
XMLRowSetName
IXMLDisplay interface 2231
XMLUseParent
TApplyUpdatesButton class 1803
TDataSetButton class 1866
TQueryButton class 1962
TXMLButton class 2156
TXMLDisplayReferenceButton class 2196
xoAttributeOnLastMember enumeration member 940
xoHolderClass enumeration member 940
xoInlineArrays enumeration member 940
xoLiteralParam enumeration member 940
xoOption6 enumeration member 940
xoOption7 enumeration member 940
xoOption8 enumeration member 940
xoOption9 enumeration member 940
xoOptionA enumeration member 940
xoOptionB enumeration member 940
xoOptionC enumeration member 940
xoOptionD enumeration member 940
xoOptionE enumeration member 940
xoOptionF enumeration member 940
xoOptionG enumeration member 940
xoOptionH enumeration member 940
xoOptionI enumeration member 940
xoSimpleTypeWrapper enumeration member 940
XSDToTypeInfo
TRemotableTypeRegistry class 891
XSToNative
TRemotableXS class 895

Y

Y
TRibbonTabItem class 3560